Report: A Short Story Writer

# Abstract

A Short Story Writer was created to produce creative pieces of text based on the architecture of Markov Chains/Processes and a corpus/language provided to the program. Markov Processes are notably reliant on probability, so to avoid generating the same output reoccurring, a ‘Fitness Function’ was implemented, having ‘weightings’ of the functions, similar to that of a value network within neural networks. This allowed the system to increase or decrease certain weight attributes such as ‘Rhyming’ or ‘Punctuation’ in order to bias the text generation to create more creative and interesting texts. The corpus used for A Short Story Writer was the summation of Barack Obamas speeches, as these provided the program with clear, error free text which it uses to generate a new story. The system was also tested on a corpus of the Aladdin story in order to generate a text resembling a story.

# Introduction

Computational Creativity has been at the forefront of discussion for the Computer Science community as it incorporates interesting aspects such as Machine Learning and Artificial Intelligence. The Artificial Intelligence (AI) field has been growing and attracting a great deal of attention among different industries and the general population for its problem-solving capabilities. For instance, uses such as automated robot arms utilise in assembly lines, or simply personal assistants like Cortana (by Microsoft) of Siri (by Apple) that are increasingly assisting humans in daily life.

Automated story generation is the use of artificial intelligence to create novel or fictional stories. These systems have been most successful when creating fictional stories where there is a finite set of actions, characters and other objects, such as automated soap operas (Lebowitz 1987). Although impressive, they only work well within the constraints and boundaries provided to them and lack the ability to produce coherent stories outside of these boundaries. As a result, there is difficulty determining whether the quality of the stories produces by automated story generation systems is due to the algorithm or engineering of the story. Similar to Automated Story Generation, Open Story Generation is the task of automatically generating stories and texts without a prior knowledge of the domain world. It can do this by either learning a domain model from data provided to it, or to use data provided to it from a corpus (Data set).

Inspiration for A Short Story Writer stemmed from previous works on this topic such as Kliens Novel Writer (1973), Dehn’s Author (1981), Mehan’s Talespin (1977), The Virtual Storyteller (2003) and Mexica (1999). These previous works are an insight into how creative text systems have been designed and implemented until now. The aim of A Short Story Writer is to generate short stories based on a corpus of speeches and stories.

This report will be detailing the background, design, development, implementation and evaluation of the Short Story Writer project. The Background section is focused on provided technical background to theories used in my program, reviewing research papers which relate to my work, as well as looking at other creative systems similar to mine and how they have fared. The Methodology and design section is focused on providing a technical introduction to project, outlining its development stages and how it was designed. Finally, I will look at the results produced by my program and evaluate it with respects to the quality of its output and in comparison, to other creative systems.

# Background

## Markov Chains

Markov chains are mathematical probabilistic automaton which describes a sequence of possible events in which the probability of each event depends only on the state attained in the previous event [2]. It’s probability distribution of state transitions is typically represented as the transitions matrix. If a Markov model has N possible states, the matrix will be an N x N matrix, such as an entry (M, E) is the probability of the model transitioning from state M to state E. Also, this transition matrix must be a stochastic matrix, which is a matrix whose entries in each row add up exactly to 1. This is necessary as each row represents its own probability distribution. Furthermore, the Markov Chain also has an initial state vector, represented as a N x 1 vector, that describes the probability of starting the chain at each of the N possible states. These two factors describe the main basis of the Markov chain model.

## Other Systems

The foundation of creative text generation for current designs and processes was laid by earlier works, such as Klein’s Novel Write (1981). This novel writer worked by simulating a model where the behaviour of individual characters and the events that transpired were governed by probabilistic rules that progressively changed the state of the simulated world [1]. Although its results were depended on strict rules set beforehand, its probabilistic nature can be found in aspects of Markov Models, which use probability to determine which word will appear after a certain word. This influenced A Short Story Writer as it allows the system to pre-assign the starting word of the Markov chain. Thus, giving the user the power to bias the story at different stages of the text in order to create a story with a coherent plot.

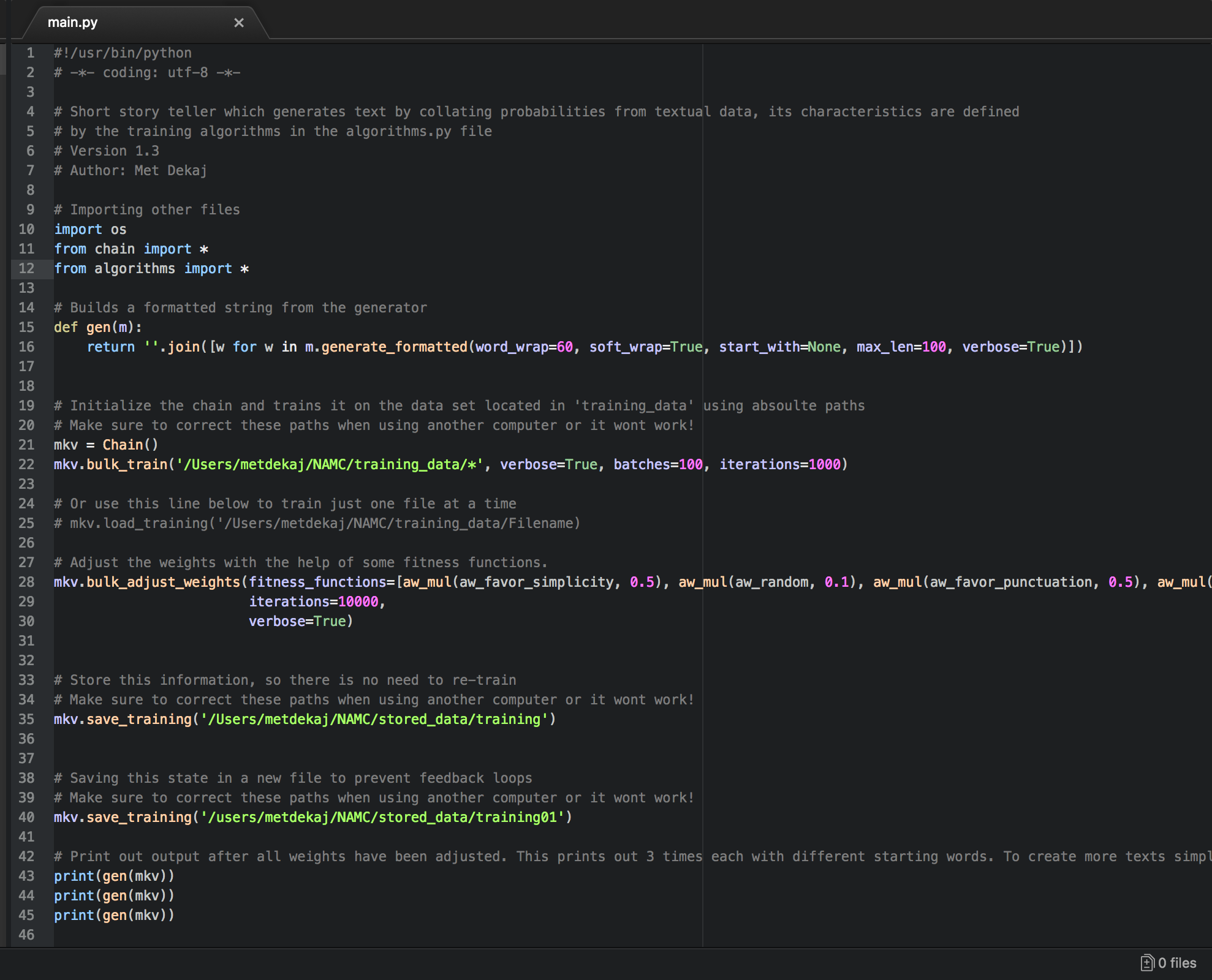
Other developers have tried to create automated story generation systems by improving upon the Markov Process model, such as the system created by Brent Harrison, Christopher Purdy and Mark O. Riedl in their paper called ‘Toward Automated Story Generation with Markov Chain Monte Carlo Methods and Deep Neural Networks’. The paper describes a system which approaches automated story generation by using a sampling algorithm based on Metropolis-Hastings (Chib and Greenberg 1995) to generate a probability distribution which can be used to generate stories via random sampling that adhere to criteria learned by recurrent neural networks [?].

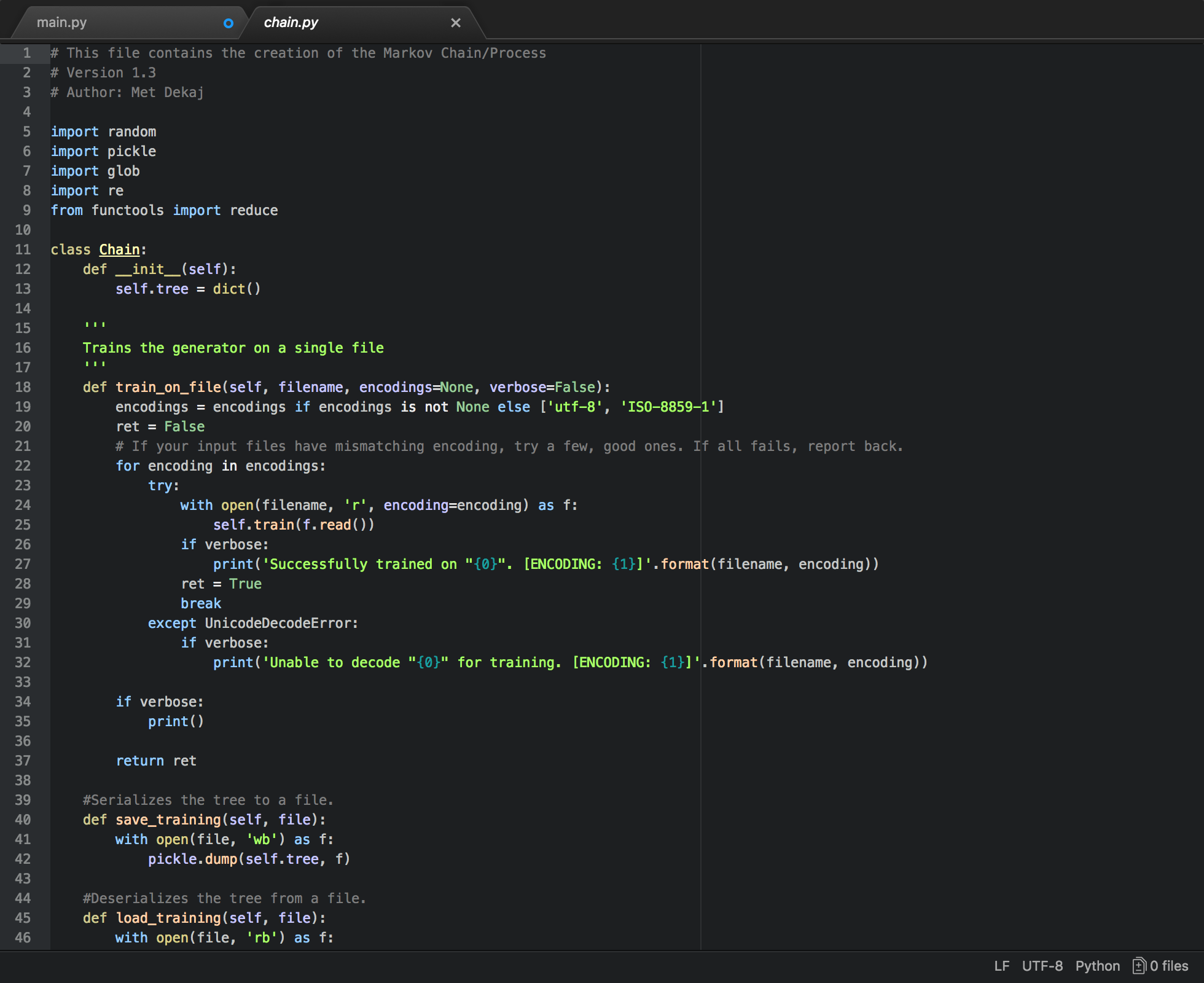
The creative system in this paper guides the story generation by using a value function f(x) which analyses each sample x drawn from a distribution table, and either accepts it or rejects the sample based on some criteria provided to the system. This allows the developers to bias the system to generate stories within certain criteria or boundaries. Although unable to create a program similar to the system in this paper, it was possible to use ideas from the criteria model from this implementation to start text generation with a word of my choice, allowing the system to bias the text in order to resemble a story.

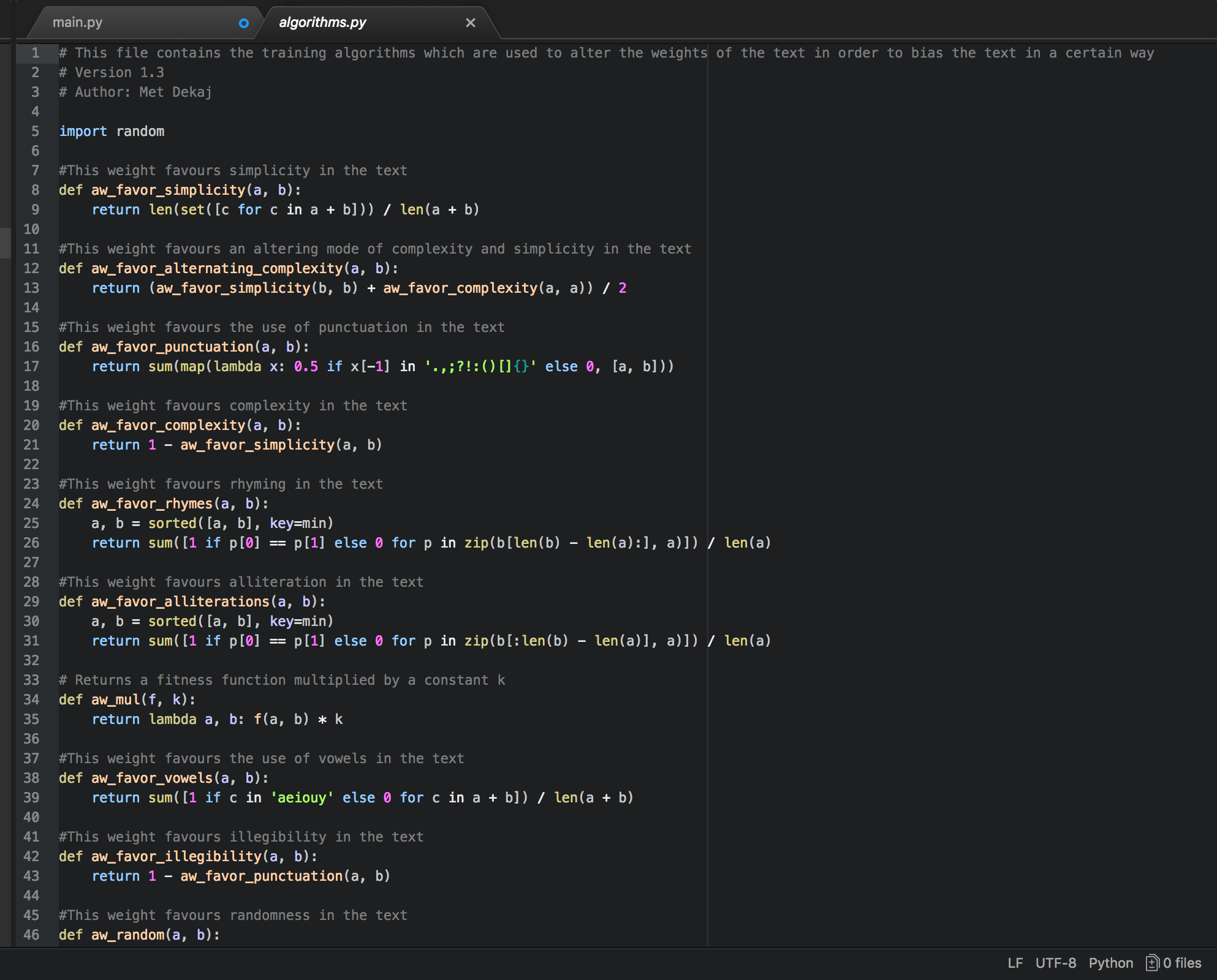
# Methodology and Design

After reviewing several story generation creative systems, it seemed that Markov Processes were among some of the most popular, and the most customizable in terms of the type of story generated. I decided to create my program based on the Markov Chain/Process. The Markov model describes a sequence of possible events in which the probability of each event depends on the state attached in the previous event. When applied to text generation, this looks at the probability of a word coming up based on the previous word to arise. The next word with the highest probability is inserted and the process is continued. Previewing the previous word is often called First Order, describing that the next word input is only calculated by looking at the previous word. This can be increased to the Second and Third Order, which calculated the probability of the next word coming up when looking at two or three words back, respectively. The trade off with increasing the order is although you may get more coherent text generation, you also lose some sense of randomness and creativity, as the text becomes more normalised. Furthermore, increasing the order requires a much larger corpus, as a program with a high order and small corpus runs the risk of generating the same text as the corpus.

I decided to create the program using Python due to its simplicity in writing and de-bugging code compared to other languages such as C++ and Java. Furthermore, python has many existing, well tested libraries which are easy to import and use especially for topics like Machine Learning. For example, pickling is an imported tool in python which is primary used to serialise and de-serialise a python object structure. In other words, it allows python objects to be converted into a byte stream in order to store it in a file or database. This makes the data easier to access, manage, adapt and share across the network. This tool allowed me to pickle the corpus into a byte stream, making it easier to train the generator on the byte stream instead of the plaintext corpus file.

This is the main.py python file which generates the creative text based on probabilities from the corpus and its characteristics are defined by the training algorithms. Its main features focus on calling upon the ‘gen’ function which lies in the chain.py file which contains the Markov process. It also initializes the Markov chain and trains it on the data set located in the absolute path. It also calls upon the fitness functions, allowing the programmer to adjust the weights of the different functions in order to produce biased texts. It also stores the information in a file so there is no need to re-train on the same corpus, it also saves the state in a new file path to prevent feed-back loops. It then complies all these functions and prints out the generate function in order to create a piece of text.

This is the chain.py file which represents the Markov process. It works by importing random, pickle, glob and re libraries. The file starts of by training the generator on a single file, checking the encoding in UTF-8 and ISO-8859-1. The file then sterilizes the tree to a file so that it is easier to read, and then desterilizes once complete. It also then trains the generator on a block of text by splitting text at every space (including tabs and newlines), removing empty entries and keeping punctuation at the end of the word contain it. This makes it easier to work with. It then trains the generator on a single file or a list of files and saves the state to a disk when finishing using glob patterns (also returning the number of files successfully parsed and trained on). Then generate function then produce a sequence of words until a dead end is reached or max length is reached. The file then adjusts relationships between branches according to the fitness functions. The file finishes off by calling the adjusted weights with the multiplied result of the fitness functions for a given number of iterations. It also shows a progress bar if the compilation has been successful.

This is the algorithms.py file which contains the training algorithms used to bias the text generation in a certain way or pattern. As shown from the screenshot, a function such as aw\_favor\_simplicity could be used with a weighting of 0.5 to bias the text generation to engage simplicity in the text more often. I have included around 10 different training algorithms, most of them in regard to the use of complexity of the text, punctuation, vowels and constants. Each algorithm is capable of having a weighting between 0 and 1 in order to customize the output effectively.

# Results

This section will consider the results produced by the system, specifically focusing on the quality of the results and the level of creativity produced. As mentioned previously, the output of the system greatly depends on the corpus fed to the system, therefore several different bodies of text have been tried in order to get a better idea of how different texts work in the system.

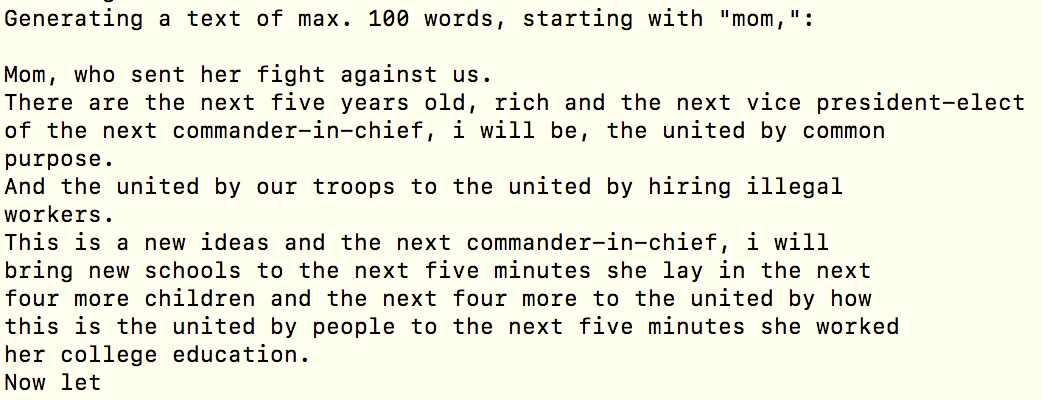
Initially, the system was trained on several different speeches made by Barack Obama due to his simplistic style. This made it easy for the system to produce understandable texts while completely differing from the initial speech made. As you can see in Figure. 1., the majority of the text does not make logical sense and does not have a feeling of creativity as it seems as though words are being outputted solely based on probability, thus giving random sentences. In order to overcome this, the weightings of the fitness functions were tailored in order to produce a more coherent and creative text.

Figure. 1. The systems output based on a corpus of Barak Obama speeches.

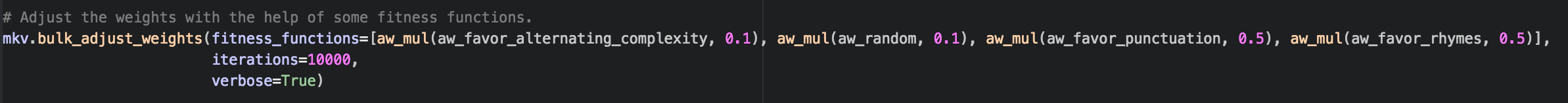
As shown in Figure. 2., small weightings were added to alternating complexity (which allows the system to sometimes go for simplicity, and sometimes go for complexity), randomness (which allows the system to add an element of randomness to its decision making), punctuation (allowing the system to favor punctuation in its decision making) and rhyming (allowing the system to choose words which rhyme more). These fitness functions allow the system not to simply go for probability but add extra elements which change the decision-making process, producing more interesting and creative texts.

Figure. 2. Display of the code that small weightings were added to alternating complexity.

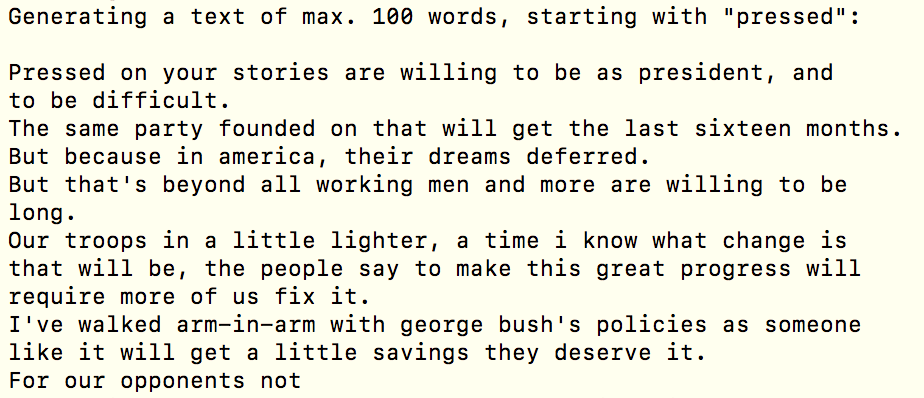
Figure. 3. shows text generation using the same corpus with consideration the added fitness functions and the weightings. You can notice a rise in the creativity of the text produced, as there is less grammatical mistakes and more fluid expressions. However, there are still sections of the output which don’t make total sense. This could be due to the added random (0.1) fitness function which was added to create more interesting texts.

Figure. 3. The systems output based on a corpus of Barak Obama speech with implemented weightings.

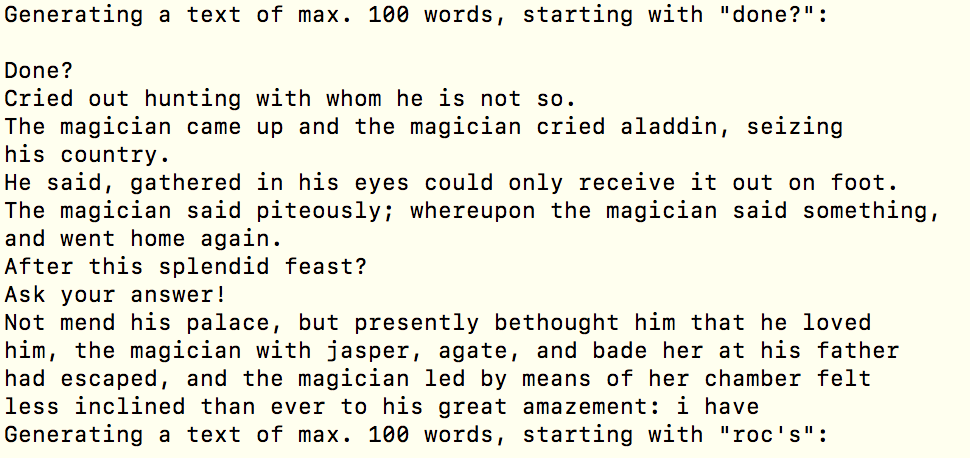
The corpus is changed to Aladdin and the Magic Lamp, in the hopes of creating a text which resembles a story. As depicted in Figure.4., the text generated does resemble more of a story, as it includes characters, questions and other events. However, it is still lacking some grammar and misplacing words. Perhaps this could be solved by changing the weightings of the fitness functions to favor more simplistic text generations, therefore it may focus on getting more accurate sentences, instead of interesting and creative sentences. At this point in time, I believe that the output by the system displays a decent level of creativity, especially when playing with the fitness functions, as it does not only rely on probability but other factors.

Figure. 4. The systems output based on a corpus of Aladdin and the Magic Lamp.

# Evaluation

I have decided to use SPECS (Standardised Procedure for Evaluating Creative Systems) as the evaluation process for this creative system as it provides a flexible system capable of evaluating most creative systems. Furthermore, SPECS covers evaluation of systems in a systematic way, as shown in the lectures notes [5], most systems are not evaluated in a systematic or standardized way, --SPECS ensured that the evaluation process is thorough. The process consists of three steps; Investigation, Standards of Evaluation and Testing.

## Investigation

This section identifies a definition of creativity that the system produced should satisfy in order to be considered creative.

### What does it mean to be creative in a general context, independent of any domain specifics?

In order to answer the question what it means to be creative, one must first define what they believe creativity means and resembles. For this I refer to Anna Jordanous definition of creativity as 14 components which provide an ontology of creativity [6]. This collective definition of creativity is defined by its subparts; Active Involvement and Persistence, Dealing with uncertainty, Domain competence, General Intellect, Generating Results, Independence and Freedom, Intention and Emotional Involvement, Originality, Progression and Development, Social Interaction and Communication, Spontaneity and Subconscious processing, Thinking and Evaluation, Value, Variety – Divergence and Experimentation.

### What aspects of creativity are particularity important in the domain your system works in?

In terms of Computational Creativity and text generation, I believe there are a few key components of the 14 components of Creativity which accurately describe the domain my system falls under. These components are; Dealing with uncertainty, Generating Results, Originality, Progression and Development

## Standards of Evaluation

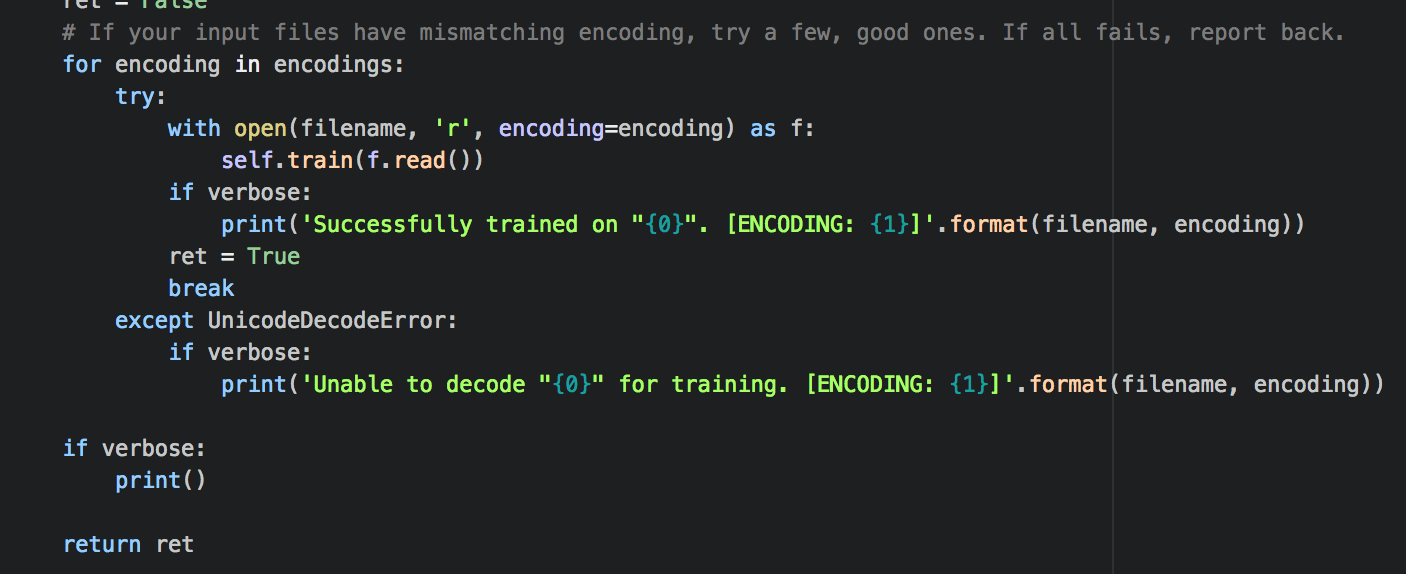
This section identifies the criteria for creativity included in the definition and extracts them from the definition, expressing each criterion as a separate standard to be tested.

* Dealing with uncertainty is not relying on every step of the process to be specified in detail; perhaps even avoiding routing or pre-existing methods and solutions. I believe this is a criterion for creativity and should be tests because if a system is not able to deal with uncertainty, then it is simply following strict rules and guidelines laid out by the programmer and not making critical decisions for itself.
* Generating Results is a producing something (tangible or intangible) that previously did not exist or working towards a goal/end target. I believe this should be tested against because if the system is not able to generate results then it would be difficgult to measure its creativity.
* Originality is the production of a new product or doing something in a new way; seeing new links and relations between previously unassociated concepts and producing results that are unusual and out of the ordinary. I believe this to be a key concept and should be tested as the system should be able to produce a body of work completely different than the corpus it was given.
* Progression and Development is when the system’s process should represent some progress or evolution in a particular task. I believe this should be tested as the system should be able to run many iterations of the same process but gain different results due to its progression.

## Testing

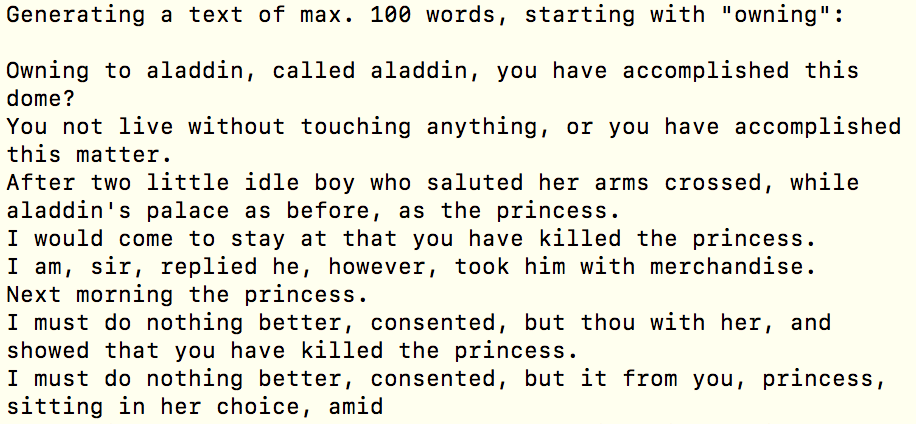
### Dealing with uncertainty

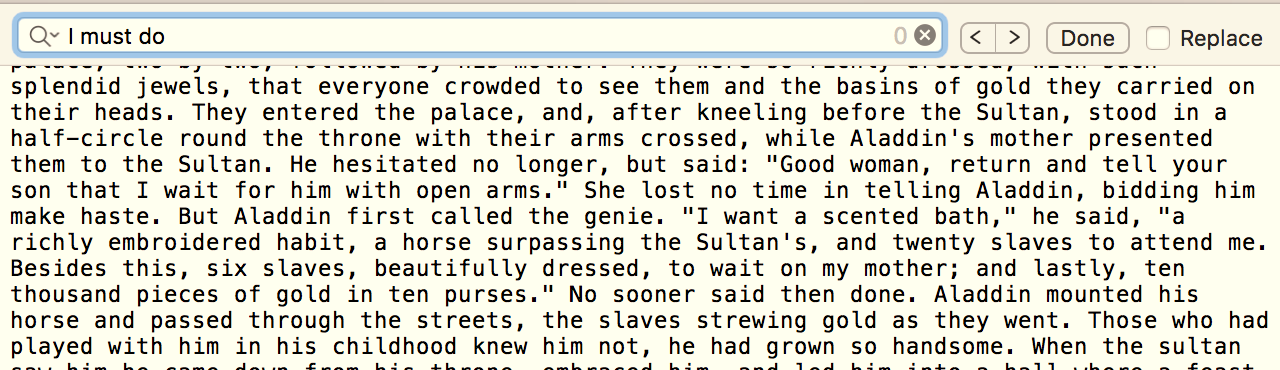
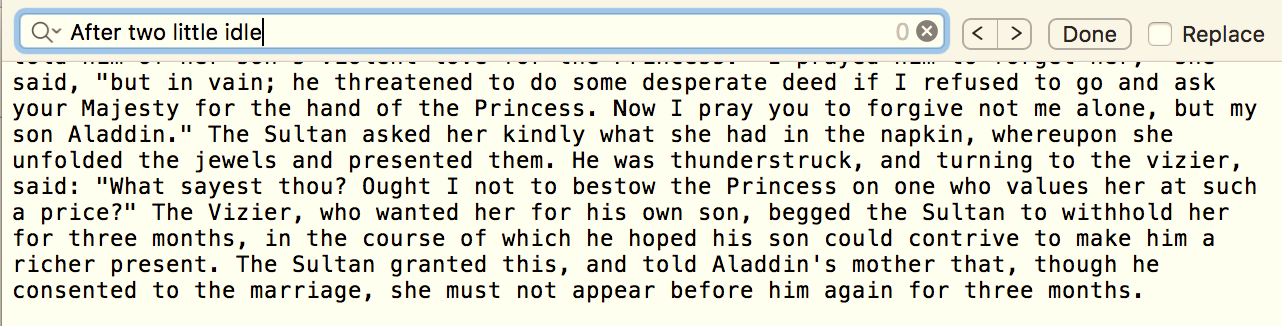
As explained, a system needs to be able to deal with uncertainty when presented with a problem in order to show creativity. I believe this is an important aspect of creativity, as

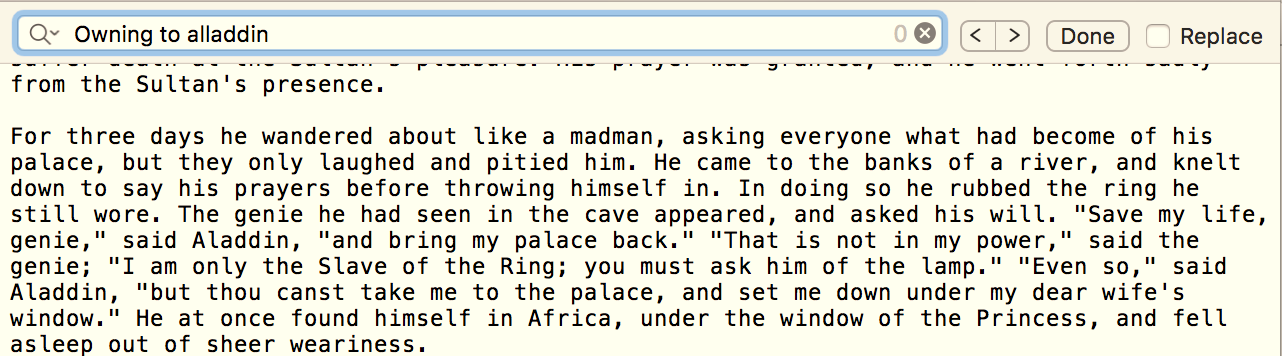
As seen from the screenshot, this is the function which trains the generator on a single file. In order to train the generator on the file, the file must be read by the system either in UTF-8 or ISO-8859-1. In certain texts, there may be characters that are unrecognisable when using either UTF-8 or ISO-8859-1, so the system checks if the character can be read by the opposite encoding, for example; if the file is being read in UTF-8 and there is an encoding that is unrecognisable, the system will try and de-code that specific character in ISO-8859-1 in an attempt to resolve the issue. This can also work vice-versa is the system is initially decoding with ISO-8859-1.

### Generating results and Originality

As explained, systems need to be able to work towards a target to produce something that previously does not exist. This can be tested in the short story writer system by comparing the original corpus fed to the system against the output it produces. By comparing these two, we can see if the system is simply repeating its input or producing something different.

This Is an extract of 100 words produced by the system when given the corpus of Aladdin and the Magic Lamp. In order to test the results provided by the system, I have looked for phrases in the output that may exist similarly in the input file. This will allow me to compare if the output is simply a regurgitation of the input or a creative outcome of system intelligence.

As seen in these screenshots, several phrases from the output have been compared to original corpus of Aladdin and the Magic Lamp, but none have proved to be similar in phrase or punctuation. From this we can infer that the system is able to generate results that have an aspect of originality.



Furthermore, the fitness function ‘Favour Randomness’ has been given a weight of 0.1 in the system. This allows the system to add aspects of randomness into its output, giving it a sense of originality as it is less likely to input the same word as presented in the corpus.

As originality is an important part of creativity, it is essential to note that failing these tests would deem the system as not creative.

## User Study

In order to gain a better understanding of the creativity produced by the system, i decided to create a user study in order to receive feedback on the output. This allows me to see from other points of view how creative the system seems.

* Ariona Kruezi rated the system 5/10, stating that “The text produced is certainly interesting and I would say also creative especially for the texts produced from the Barack Obama Speech as they flow much better, however there are still gaps that require us to fill in as the text generated does not make sense in those sections”.
  + I agree with Ariona’s evaluation of the system as in some sections, there is clear understanding with what the system has produced, and in other sections it is more difficult to comprehend what is being said. I also agree with the fact that text generation using the Barack Obama speeches were more coherent, perhaps due to the simplicity of the speech.
* Abdur Rahman rated the system 6/10, stating that “The text generated from Aladdin was very funny and unexpected, probably funnier than the actual story, the Barack Obama speeches were also interesting, but both could have benefited from more punctuation and grammar to make it sound more real”.
  + Abdur’s evaluation makes a good point about increasing the use of punctuation in the speech to make it sound better, using grammar would also be an interesting concept, especially if context-free grammar ideas were incorporated into the Markov chain.

# Conclusions

In conclusion A Short Story Writer is capable of producing strings of text that form a story of logical sense, with an aspect that users may view as creativity. By using the Markov Process in conjunction with Fitness Functions, this allows the system to bias the text generation in order to form a more coherent story. In the future, the system could be improved by combining the Markov process and fitness functions with Context-Free Grammar methods. This would generate much more coherent stories based on a plot which includes a beginning, climax and ending. A further improvement upon this would be to use pair the Markov process with Monte Carlo searching algorithms and Deep Neural Networks to improve efficiency and accuracy. Also, a minor improvement which could not be implemented due to timing constrains was to start every new sentence with a newly generated word, allowing me to bias the word at each stage of the text in order to resemble more of a story. Overall, the system seems to produce creative pieces of text given an appropriate corpus, however it is clear that certain aspects of the algorithm could be improved upon.

# References

[1] Gervas, P, 2009, ‘Computational Aproaches to Storytelling and Creativity’ , AI Magazine, Vol 13, No.3

[2] Oxford Dictionaries | English. (2018). Markov chain | Definition of Markov chain in US English by Oxford Dictionaries. [online] Available at: https://en.oxforddictionaries.com/definition/us/markov\_chain [Accessed 4 Aug. 2018].

[3] Harrison, Brent, Christopher Purdy, and Mark O. Riedl. "Toward automated story generation with markov chain monte carlo methods and deep neural networks." Proceedings of the 2017 Workshop on Intelligent Narrative Technologies. 2017.

[4] Sierra, Basilio, and Elena Lazkano. "Markov Text Generator for Basque Poetry." Text, Speech, and Dialogue: 20th International Conference, TSD 2017, Prague, Czech Republic, August 27-31, 2017, Proceedings. Vol. 10415. Springer, 2017.

[5] W15\_1+2evaluation\_of\_creativity, Page 11 –Evaluation of creative systems: A review of the situation in 2010.

[6] Jordanous, A. and Keller, B. (2016). Modelling Creativity: Identifying Key Components through a Corpus-Based Approach. [online] Journals Plos. Available at: http://journals.plos.org/plosone/article?id=10.1371/journal.pone.0162959#sec011 [Accessed 4 Aug. 2018].