„FAIRYTALE GONE BAD“

The Fairytale Generator

By Sophia Mahnke

The topic of my final project, generative literature, belongs to the field of artificial or computer generated art. The term “generative” is often used in the context of producing and creating, especially in combination with processes that are in some way automated by a machine or computer.

Generative art occurs in different forms, including not only sound and visuals like imagery or videos, but also pieces of literature, such as poetry or other kinds of stories.

Philip Galanter defines generative art as a practice in which “artists use systems like computer programs, machines or other procedural inventions” that achieve autonomy (P. Galanter, “Generative and Rules Based Art”, 2006).

In contrast to the beginning of the computer era, in which people were afraid of computers erasing human creativity, the attitude has changed, is more harmonic and computers are used as tools that enhance the understanding of creativity. Generative literature is a specific form of digital literature is based on specific dictionaries, some set of rules and the use of algorithms. These literary pieces are produced by the computer, which is why they do not have an actual, human author. Characteristics of generative art and literature are that every piece is unique and non-repeatable, the results are unpredictable and each emerge with different degrees of autonomy.

There are different kinds of generative literature. On the one hand, there are story generators that either mix together a selection of stories and texts or make a story around given user input. On the other hand, generative literature can also be co-authored poems or stories. In this case the program uses predictive text and gives the “author” a few words to choose from.

In the context of generative literature, the topic of artificial intelligence and the question whether machines/computers are able to think come into mind. Alan Turin came up with the question in his paper “Computing Machinery and Intelligence” in 1950, in which he states that if the computer is able to win the “imitation game” (Turing Test) , it is capable of thinking like a human. The basic Turing Test has one questioner and one human and one computer that respond. It there are correct determinations half of the tests or less, the computer has artificial intelligence, since the questioner regards him as a human.

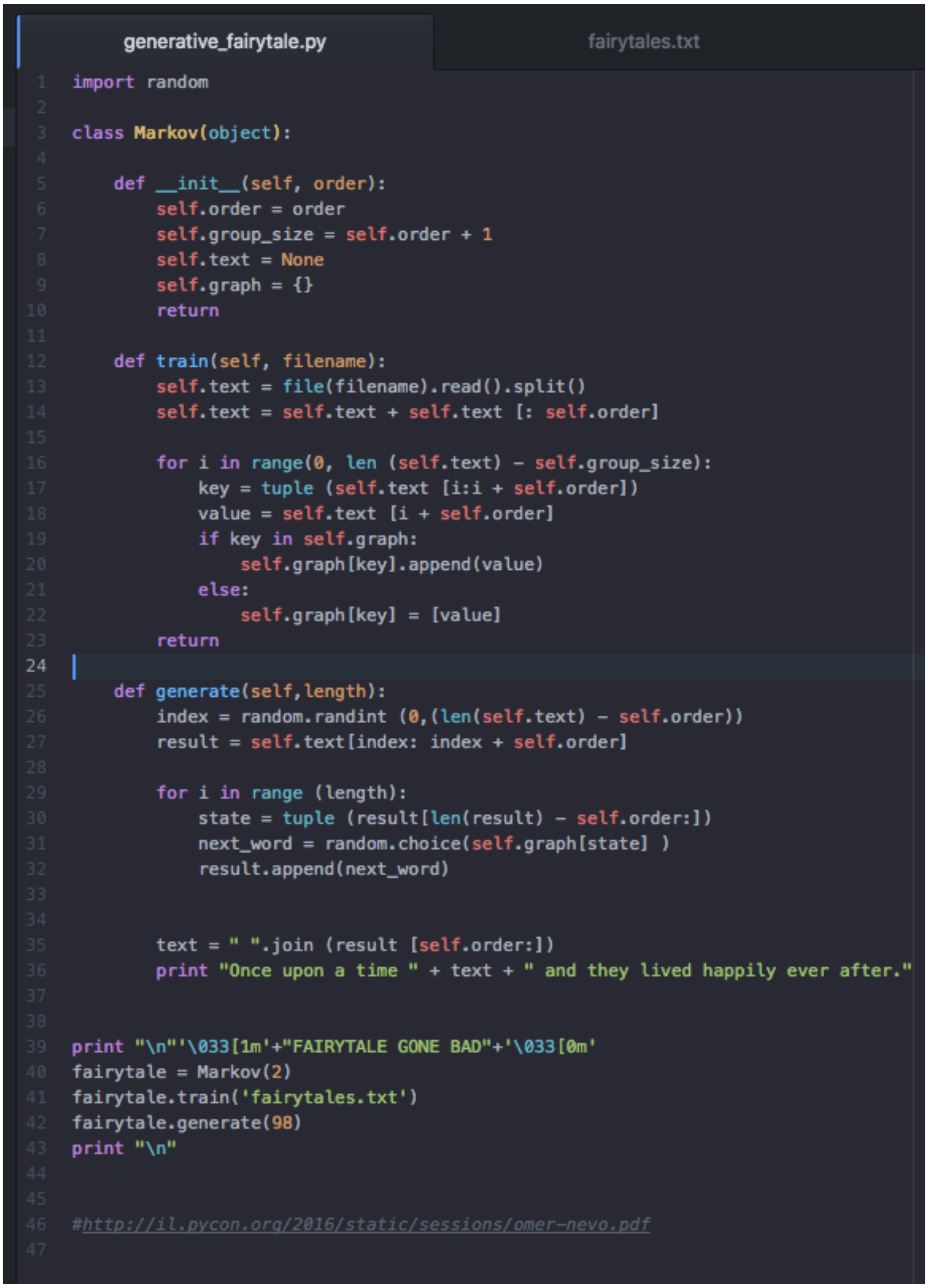
The next question in this context is whether computers can write literature: yes, they can – even though the literary pieces will not make much sense.

An example for generative literature is the Harry Potter book generator by Robert Clouth. The title of this project is “Harry Potter and the Statistically Generated Nonsense” which perfectly puts the whole project into a nutshell. The user clicks on a button to make a new book and receives ten randomly generated chapters with word or lines of the Harry Potter books.
What interests me the most are the points of autonomy and automation, authorship, uniqueness and creativity.
Inspired by the Harry Potter Generator, my final project will be a fairytale generator which mixes together commonly known fairytales and creates a new, funny fairytale (which does not make sense).
The basis of generative literature is the Markov chain, which is a method from probability theory and a way to model reality. For example, you can model the weather with the chance of rain. In the case of literature, it is a Markov model on words in which the program identifies, how likely it is that one word follows another specific word or a word pair.
I already wrote a code, with the help of an instruction by Omer Nevo, which generates a random fairytale.
An example of a randomly generated fairytale is this:

```
FAIRYTALE GONE BAD
Once upon a time princess, open the door to, in great grief and sadness. The
carriage was to conduct the young King into his kingdom. Then they went to the
little hand which she had placed herself beneath this, and cried, "That is the
true one, and that the tears fell down he was terribly frightened when a man
and a pickaxe that he was young and handsome, she thought, "He will love me and
let me be thy companion and play-fellow, and sit by thee at thy little golden
plate, something came creeping splish splash, up the marble staircase, and when
and they lived happily ever after.
```

The database for this fairytale are Brother Grimms’ “The Frog King”, “Cinderella” and “Rapunzel” and I put the “Once upon a time” at the beginning and “and they lived happily ever after” at the end so that the generated “fairytale” gets the fairytale character.
The next steps I am going to take are:

1. Increase the database, include more fairytales
2. Edit the code, trying to get it a bit more “structured” and that the words fit with the beginning
3. Create website, on which users can generate fairytales



```python
import random

class Markov(object):

    def __init__(self, order):
        self.order = order
        self.group_size = self.order + 1
        self.text = None
        self.graph = {}
        return

    def train(self, filename):
        self.text = file(filename).read().split()
        self.text = self.text + self.text [: self.order]

        for i in range(0, len (self.text) - self.group_size):
            key = tuple (self.text [i:i + self.order])
            value = self.text [i + self.order]
            if key in self.graph:
                self.graph[key].append(value)
            else:
                self.graph[key] = [value]
        return

    def generate(self,length):
        index = random.randint (0,(len(self.text) - self.order))
        result = self.text[index: index + self.order]

        for i in range (length):
            state = tuple (result[len(result) - self.order:])
            next_word = random.choice(self.graph[state] )
            result.append(next_word)


        text = " ".join (result [self.order:])
        print "Once upon a time " + text + " and they lived happily ever after."


print "\n"'\033[1m'+"FAIRYTALE GONE BAD"+'\033[0m'
fairytale = Markov(2)
fairytale.train('fairytales.txt')
fairytale.generate(98)
print "\n"


#http://il.pycon.org/2016/static/sessions/omer-nevo.pdf

```

References:

- Alan Turing (1950) Computing Machinery and Intelligence, **https://www.csee.umbc.edu/courses/471/papers/turing.pdf**
- Philip Galanter (2006) Generative Art and rules-based Art, **http://philipgalanter.com/downloads/vague_terrain_2006.pdf**
- **http://il.pycon.org/2016/static/sessions/omer-nevo.pdf**
- **http://www.soban-art.com/definitions.asp**
- **http://www.digitalpedagogylab.com/hybridped/what-is-generative-literature-introducing-the-generative-literature-project/**
- **http://www.dichtung-digital.de/2005/1/Balpe/**
- **http://setosa.io/ev/markov-chains/**