# Logbook for Poem Generator

Egon Janssen (s1530100); Unmukt Deswal (s2310171) 04/11/2020

## Implementation & Motivation

This is a logbook of our implementation where we describe a step-wise account of the tasks that we implemented as a part of this project. We introduce the motivation for this project (which was not merely scoring well in this assignment), followed by describing our Inspiration Set and motivations behind choosing it. This is then followed by discussing generation and language model that we implemented and finally discuss the transformations we made in order to attain a complex final product for generating poems. But before we do that we would like to emphasize that this project has served us well as neither of us have much experience with text generation or have worked in this domain before this project. The whole process has been truly educational and exciting and did make us re-evaluate what creativity means for such a system.

### Inspiration Set

The first step of the process was to be able to create an Inspiration Set for our poem generator. We explored a bunch of resources for our Inspiration Set and finally settled on 'Project Gutenberg corpus'. The Project Gutenberg electronic text archive was famously the first free online book provider, which contains about 60,000 free electronic books. However, our motivation behind selecting this set was simple, as it comprised of a huge dataset which accounts for big sample space for experimentation. We also noted that it was possible to select a single author's work and use feed in our Inspiration Set. This gave us a sense of assurance, that consistency in writing style was somewhat retained.

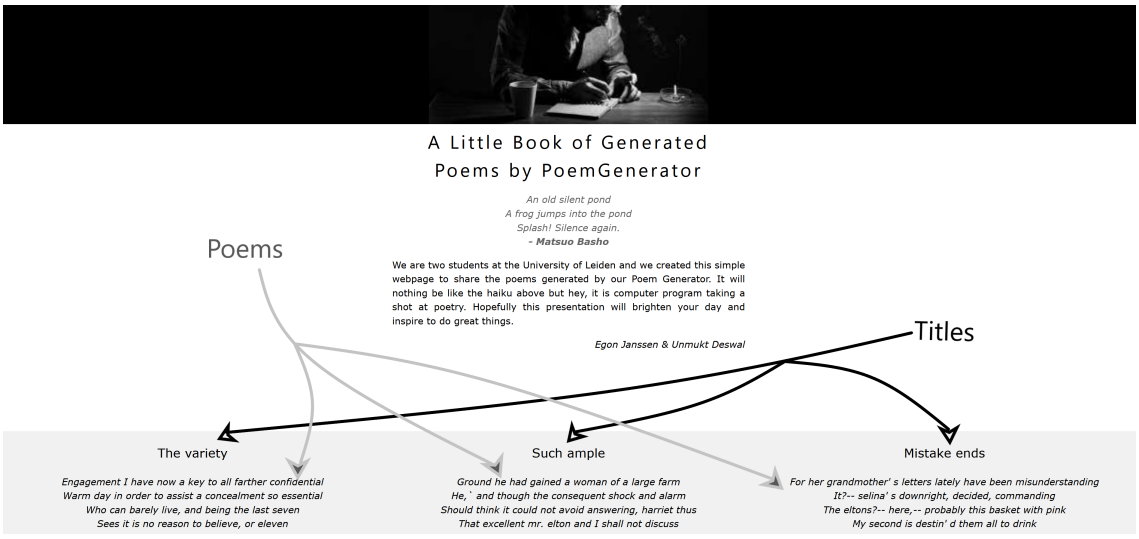
### Generation & Language Model

We weren't entirely sure about what to expect when we started with this project, and thus, a simple implementation of a text generator with a good Inspiration Set was the target. For this, we decided on implementing n-grams for our poem generator as it is the most popular way of sensible text generation. We experimented a bit with n-grams, but bi-grams and tri-grams gave the most promising results as expected, both in terms of making coherent and sensible lines. Thus, we decided to experiment with these n-grams and refer to them as couplets in our implementation. In order for us to be able to use n-gram effectively, we defined a context dictionary

in the *setupModel()* function, which returns our Inspiration Set in the form of a dictionary. This dictionary comprises of the frequency of occurrence a particular n-gram in the data. We update our n-gram based language model by feeding each individual sentence from a book one after another and update the dictionaries with information corresponding to our n-grams by using TF-IDF. In our experiments with bi-grams and tri-grams, we found that bi-grams produced results that were more promising as they made most sense for poems, since they aren't particularly meant to be verbose.

## Transformations & Generation Odyssey

As a result of successful implementation of this simple poem generator using bi-grams, we were challenged to think out of the box to add complexity and nuance. So, we decided to implement rhyme patterns in our generator to resemble true poetry for better comprehension as poetry. This was done by calling for the *findRhyme()* function that found rhyming words from the Inspiring Set. Finally, we also added text capitalisation and use of special characters such as the semi-colon(;), the comma(,) and the hyphens(-) for added complexity. All put together, it collectively brings some structure to the generated poems, thus replicating some poetic structure. Keeping all these considerations in mind, we best believe that our Poem Generator models an Inspiration set based synthesis and thus should classify in a filtration generation odyssey. We discuss some more details about the resulting poems of our generator in the concluding section. The following figure is a snapshot of our final product of our Poem Generator.



## Logs

The following is a simple log of the step-wise sequence of tasks which were carried out as a part of the overall implementation.

| Date | Time | Person | Remark |
|---|---|---|---|
| 28/10/2020 | 2 hrs | Egon | Researched resources and techniques to generate poems. Finalizing NLTK library as source. |
| 29/10/2020 | 1 hr | Egon | Implemented n-grams of NLTK corpora from Gutenberg Selections corpus as our Inspiration Set. |
| 29/10/2020 | 2 hrs | Egon | Implemented a line generator for our Inspiration set. |
| 29/10/2020 | 3 hrs | Egon | Implemented rhyming a couplet(Bi-gram). Adding a rhythm of two rhyming lines to this couplet. |
| 29/10/2020 | 1 hr | Egon | Added a title generator for poems. |
| 29/10/2020 | 1.5 hrs | Egon | Initialized the 'Little book of generated poems' as an html webpage. |
| 30/10/2020 | 2 hrs | Unmukt | Got up-to-speed with the code progress and implementation. |
| 30/10/2020 | 0.5 hr | Egon | Re-structured and concreted the project. |
| 30/10/2020 | 0.5 hr | Egon | Setup the README file. |
| 30/10/2020 | 1 hr | Egon | Converted the python files into executables. |
| 30/10/2020 | 1 hr | Egon | Cleaned up the code. |
| 02/11/2020 | 1.5 hr | Egon | Update the output html. |
| 03/11/2020 | 3.5 hrs | Unmukt | Setup Logbook. |
| 03/11/2020 | 0.5 hr | Unmukt | Update and polish README file. |
| 04/11/2020 | 1 hr | Unmukt | Finalize the logbook and prepare all deliverables. |

## Discussion & Conclusions

We believe that this poem generator is a serious contender in a race to the podium for poem generators (of course the ones made by the participants of this course). We believe that our poem generator takes into account a few things that helps it qualify as a good one. Firstly, this implementation uses bi-grams in the basic text generation process, which ensures that sensible texts are produced. Secondly, the ability of the generator to use two rhymes in a rhythmic fashion adds some kind of a poetic character to the generated texts. Thirdly, the consideration of special characters and capitalisation makes it unique and add to the overall presentation of our poems. However, we cannot say that it could ever be match up to a human in any way possible. Primarily because how our generator doesn't take into account any perceptions from the environment or even use a knowledge base per say to create these poems. Our generator could only implement most of these ideas on the basis of a good Inspiration Set. Despite having a sense of context from a good Inspiration Set, it must not be deemed as being capable of producing truly creative

poems. Which is to say, that even though we could classify this poem generator as adopting a 'Filtration' based generation odyssey, there is still much debatable about what how much this perception truly describes creativity in poetry. However, we could say that such a system would be best creative in expressing creativity in how the generated poems are structured. With that being said, having to improve on our language model, we would certainly believe that this system could introduce some interesting combinations of rhymes and alliterations. Maybe these don't introduce novelty but surely contribute to a new narrative, who knows until it's still not on paper. ;-)

. . . . . .