Generating Poetry via an N-gram Approach

Tyler Wengert
UCCS
twengert@uccs.edu

Abstract

Poetry generation has been a part of Computer Linguistics and Natural Language Processing (NLP) for a long time, and has grown and changed with the field.This paper discusses the challenges and results of an exercise in N-gram based poetry, one of the earliest forms of automated text generation.

1 Introduction

The basic idea behind N-grams is that the computer looks at an n sized group of tokens to determine the possible choices and their respective probability of being chosen next. The general consensus is that small n-grams (unigrams, bigrams), don't capture sufficient information to get a good sense of the corpus, and that large n-grams (5-grams, 6-grams), capture so much information that they begin to just output the original corpus word for word. With this in mind, I chose to take a trigram approach for my generator in order to keep some originality in the poems while still maintaining as much of the content and context from the original corpus as possible.

Of course poetry generation has advanced a good deal past N-grams, with cutting edge techniques favoring Long Short-Term Memory Recurrent Neural Networks and Transformers. With these mechanisms, computers are able to learn more about the proper structure of a sentence, make connections between sentences regarding context, and get more information about the meaning of a sentence as a whole. Poetry generated using these types of systems has been shown to be of significantly higher quality to N-gram based approaches.

2 Related Works

N-grams were, for a while, the preferred method for generating text; Galley et al. (2001) showed how the n-gram approach could be combined with the corpus methodology to produce text. Masataki and Sgisaka (1996) improved on the basic n-gram scheme by including variable n-grams to help keep greater contexts in portions of the text. Potamianos and Jelinek (1998) established that n-gram schemes outperformed decision tree schemes, n-gram approaches would continue to be used until the emergence of machine learning and neural networks. Sutskever (2011) was one of the earlier papers on the subject, and established Recurring Neural Networks (RNNs) as the standard for text generation. Xie (2017) provides a general guide to language modelling and generation with RNNs, as well as fixes for many common problems when working with this type of system. It is an excellent starting point for researchers to jump into the new and exciting parts of text generation without having to worry too much about problems with their Neural Network that have already been solved.

3 Implementation

The entirety of this program was built in Python 3.7 using the IDLE IDE. The program includes several general functions, including a switch to control the current language and a method to read in the corpus files. Some of the key functions for the generation of the poetry are explained in more detail below.

3.1 Tokenization

Tokens are the individual units that the computer uses for comparison and probability calculation. Words were used as the tokens for this system instead of characters, following from (Choi, 2016) who concluded that word tokens produced better results than character tokens.A regular expression search was used to tokenize the corpus, and included punctuation as individual tokens, rather than lumping them with their proceeding word. This seems to be the standard across many papers that explore text generation, and so it was maintained here.

3.2 Building the Trigrams

In the trigram scheme, the previous two words were used to determine the third. To build the trigrams, a nested Python Dictionary was implemented. Once the input file had been tokenized, the Trigram function grabbed the current word as well as the two following, and put them through a series of if-statement tests to see if the word and its combinations existed already in the dictionary; adding to its count if it does, or adding any entry if it does not. The final dictionary has the form:
'word': {'next_word: {two_word: 1, two_word: 3}, 'next_word': {two_word: 7}} where word is the first in a sequence of three, next_word is one of the possibilities following word, and two_word is one of the possibilities given the word and next_word. The numbers in the deepest layer indicate the count of the specific trigram.

3.3 The Next Word

Picking the next word to generate is the most important function for the generator. The system needs to pick the next word based on the previous two, so first it grabs the last two words that it output, $\mathrm{w}_{\mathrm{i\ -\ 2}}$ and $\mathrm{w}_{\mathrm{i\ -\ 1}}$, and search the dictionary. This gives it a list of all the possible words w_{p} that could be next. The system then weights each trigram so that it occurs in the list as many times as it occurs in the corpus. Therefore, a specific word P_{c} has a probability to be chosen of

$$\frac{occurences[P_{\mathrm{c}}]}{\sum_{i=0}^{size(P)-1} occurences[P_{\mathrm{i}}]}$$

3.3.1 Challenge: Starting the Poem

As explained above, the algorithm for choosing the next word is dependent on the last two words of output. This causes a problem at the start of the program because there are obviously no previous words to work with. If the system had been setup to use unigrams, it could have selected a random word from the corpus to start with, but with two words needed to get the program going, it would frequently generate two words that never co-occurred in the corpus and so the program would still get stuck. To remedy this, the system was given two seed words for each language. The chosen words were very general words, and the dictionaries were examined to find seed words that gave the generator as many options as possible to start. For example, the two seeds for English were 'to' and 'the', while the seeds for Spanish were 'un' and 'en'.

4 Evaluation

To evaluate the poems, 3 categories were defined in which a poem must score well to be considered "good": readability, content, and emotion. Readability concerns the structure of the poem. Does it have proper grammar? Do the sentences flow correctly? Do adjacent words make sense together? Content deals with the meaning and context of the poem. Does the poem keep a consistent context (ie. it talks about boats and the ocean for the entire duration) or does it mix unrelated contexts? Does the poem give more meaning as a whole than the words on their own? Finally, following from content, emotion deals with the response to the poem. Did the poem elicit emotions from the reader? How intense were those emotions?

To gather results on my poems, a short test was written that presented subjects with sample poems from the various languages and data set sizes, and asked them to rate each poem on a scale from 1 - 5 in each of the three categories. The overall score for a poem is the average of its three scores. Subjects were given a brief written explanation of the categories, but received no information about the size of the corpus from which each poem was generated. The subject pool was my roommates.

Subjects were presented with poems in English, as well as the three other languages for which reliable translations to English could be found: Spanish, Hindi, and Ukrainian. Subjects were given 1 poem from each corpus size, from each language. In total, they each scored 12 poems.

5 Further Work and Improvements

This is a fairly simple example of poetry generation, and there are several ways in which the results could be improved. The largest consideration is the N-gram scheme that was used. It is possible that a quad-gram setup could've produced better poetry. To implement this, the Trigram() function would need to be modified to include another if statement to capture the extra word of information. The next_word() function would also need to be modified to search three words back instead of just two. To further improve readability of the generated poem, more conditions could be added for the tokens, such as requiring that the first token of a line must be a word, not punctuation. To improve the start of the poem, a bigram scheme could be implemented in addition to the trigrams, so that the system could generate one random word to start, and from the bigrams get the following word, then initialize the trigram system with those two, instead of using seed words.

6 Results

Subject	Language	1k	500	200
Josh	English	3.0	2.6	3.0
	Spanish	3.0	2.6	2.6
	Ukrainian	3.0	2.6	2.6
	Hindi	2.6	3.0	2.6
Ryan	English	2.6	2.6	2.6
	Spanish	3.0	2.3	2.0
	Ukrainian	2.6	2.6	2.0
	Hindi	2.6	3.0	2.6
Luke	English	2.6	3.0	3.0
	Spanish	3.0	3.0	2.6
	Ukrainian	3.0	2.6	2.3
	Hindi	3.0	2.6	2.3

Table 1: The total score, averaged from the scores in the three categories, for each poem.

7 Conclusions

There are a number of key patterns that arise in the poem scores. Firstly, the variation in scores for a single subject between the different languages is low, meaning that the program performed approximately equally across all the languages. This is in line with the results of (Tucker, 2019). Secondly, a general trend can be seen among all the languages showing that the 1000 entry corpus produced better poetry. There are some notable exceptions to this trend; the English poem generated from the 200 entry corpus was scored the same or higher than the 1000 entry English poem by all of the participants. This is probably due to the fact that the small corpus, with less trigram options to work with, can sometimes get locked into a unique trigram sequence and output sections nearly verbatim from the corpus. The 1000 entry corpus poems tend to be significantly more original than the poems generated from smaller corpora. Some of the subject bias can also be seen, Ryan generally scored the poems more strictly then the other two participants. It is unlikely that this affects the results poorly, because the variations within Ryan's score set are very similar to the variation within the other two subjects' scores.

8 Sample Poems

8.1 English

8.1.1 1k

it made us all with perfect nonchalance ! to be solv'd , the dnieper , the sad noises of hatred and smoke of my whispering soul to himself , unfold himself , the camp of georgia and the counting ; such from one he lov'd might secretly be indifferent to

8.1.2 500

earth does not emulate those of the streets ! i am ready to start in the present to the muzzles of guns with galloping cavalry , moving with steady motion , swaying to and fro singing her husky song , as one encountered gentlefolk upon a new purchase , scorch'd

8.1.3 200

the well - possess'd of themselves . the wind does , working like a slumbering woman and her grand - daughters spun it with foot and dragoons , and who most cautious ? for i am august , i understand you , and favor men and women die ! a

8.2 Spanish

8.2.1 1k

instante que yo ame a dios y a su amada , de llevar la mañana , la portera . de dónde vino . . víctima condenada a vegetar sumida en un bosque , calla . tendida tú aquí . sobre tierras y mares su horizonte es mi ilusión sobre el

8.2.2 500

donde tu alma en el crepúsculo negro en tus atribuciones cuando el artista tiene un arma irresistible para ellas el desprecio a los hombres , mujeres . niños como una cintura . un corazón quemado y desde ese día tú no sabes ni adivinas la suprema ventura que se levantó

8.2.3 200

dulce sabor a violines , donde tu alma miré hasta su interior . rotas estaban tus virgíneas alas que nos ha reunido , no fue la tierra y en cambio sólo te exijo un cariño para mí . ay , si acerco mi boca palpita , seno de amor libre

8.3 Ukrainian

8.3.1 1k

землю знову тінь й не знати братці добрий він чи ні нам хуани - марьї ось де по війні ? засмакує сало як колись стало зимно без імперії комусь , ті про когось іще , вже по тій руді . ти , сказав собі зажди . невже до нас руки

8.3.2 500

людей а побачив дощ що мляво витанцьовував куполами парасоль під якими перехожі ховали від мене мало , не так уже пьотр велік бідна кляра від горя плаче ні намиста ні перемог ще й телескоп . і все те у сталій свідомості , що справа в тому , та менше з

8.3.3 200

і шафи , і множаться , довшають тіні . от і добре , що еверест за гратами , в діло пішли каменюки й дубини . вороне чорний ! даремно ти крячеш ! ріднеє слово жиє і є бачиш ? рідна моя , ти сучасного мистецтва субститут . і тверезу холодну

9 Acknowledgements

There are several papers that I read to help me with this project that I did not directly reference. For n-gram models: (Jelinek, 1985), (Kurzweil and Keklak, 2007), (Fillmore). For RNNs/LSTMs: (Bena and Kalita, 2020), (Xie et al., 2017), (Potash et al., 2015).

References

Brendan Bena and Jugal Kalita. 2020. Introducing aspects of creativity in automatic poetry generation. arXiv preprint arXiv:2002.02511.

Fazekas G. Sandler M Choi, K. 2016. Text-based lstm networks for automatic music composition. arXiv preprint arXiv:1604.05358.

Nathanael Fillmore. A romantic poetry generation.

Michel Galley, Eric Fosler-Lussier, and Alexandros Potamianos. 2001. Hybrid natural language generation for spoken dialogue systems. In Seventh European Conference on Speech Communication and Technology.

Frederick Jelinek. 1985. Markov Source Modeling of Text Generation, pages 569–591. Springer Netherlands, Dordrecht.

Raymond Kurzweil and John A Keklak. 2007. Basic poetry generation. US Patent 7,184,949.

Hirokazu Masataki and Yoshinori Sgisaka. 1996. Variable-order n-gram generation by word-class splitting and consecutive word grouping. In 1996 IEEE International Conference on Acoustics, Speech, and Signal Processing Conference Proceedings, volume 1, pages 188–191. IEEE.

Gerasimos Potamianos and Frederick Jelinek. 1998. A study of n-gram and decision tree letter language modeling methods. Speech Communication, 24(3):171–192.

Peter Potash, Alexey Romanov, and Anna Rumshisky. 2015. Ghostwriter: Using an lstm for automatic rap lyric generation. In Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 1919–1924.

Martens J. Hinton G. E Sutskever, I. 2011. Generating text with recurrent neural networks. Proceedings of the 28th international conference on machine learning (ICML-11), pages 1017–1024.

Shaun C Tucker. 2019. Transfer learning across languages with poetry.

Stanley C Xie, Ruchir Rastogi, and Max Chang. 2017. Deep poetry: Word-level and character-level language models for shakespearean sonnet generation.

Ziang Xie. 2017. Neural text generation: A practical guide. arXiv preprint arXiv:1711.09534.