NLP based Poetry Analysis and Generation

Technical Report · December 2016

DOI: 10.13140/RG.2.2.35878.73285

1 author:

Marmik Pandya
Northeastern University
1 PUBLICATION **2** CITATIONS

SEE PROFILE

Poetry Analysis and Generation

Marmik K Pandya

Northeastern University

Abstract

What makes a particular poem better than another or a particular poem, poet's masterpiece? There are various mathematical models driven by Rhymes, meter, style, affect, imagery et. Al. proposed for answering the same question. Now, the question arises if there exist a computational model to analyze poetry, then can computer generate poetry? Natural Language Generation is a popular research field in NLP and there are many approaches right from using Markov chains and Context Free Grammar to using recurrent neural network and other deep learning techniques proposed to make computer generate poetry. This study tries analyze a few computational models been suggested for analyzing poetry and also discuss a few of the poetry generation approaches.

Poetry Analysis and Generation

Poetry is nerved with ideas, blooded with emotions, held together by the delicate, tough skin of words.

—Paul Engle (1908 -1991)

Millions a taken aback by awe-inspiring beauty of poetry and yet there's just a little bit of empirical research done in developing computational models for poetry generation. Among the research that has been done. Research that has been done in the field includes quantifying poetic devices such as rhyme and meter (Hayward, 1996; Greene et al., 2010; Genzel et al., 2010), tracking stylistic influence between authors (Forstall et al., 2011), classifying poems based on the poet and style (Kaplan & Blei, 2007; He et al., 2007; Fang et al., 2009). Although there's been a little work done on building frameworks based on rhyme, meter, word-stress patterns et. al., there's a little or no work done on analyzing poetry based on aesthetic beauty of the words. Little research done in the area includes classifying the quality of poetry based on Style, Affect, and Imagery(Kao and Jurafsky 2012) et. al.,

The infinite monkey theorem states that a monkey hitting keys at random on a typewriter keyboard for an infinite amount of time will almost surely type a given text, such as the complete works of William Shakespeare.

- *Infinite Monkey Theorem*

On contrary to poetry analysis, Natural language generation, is a highly active research area under Natural Language Processing. There's been a lot of approaches proposed for poetry generation with a few including viewing poetry generation as a schotastic space search problem and using hill climbing to solve this problem (Manurung et al. 2000), using a genetic algorithm

to approach the search problem (Manurung et al. 2009). Apart from these other corpus-based methods include, extracting emotional content using sentimental analysis from a corpus of text and then using those tags to generate poetry based on rhyme and meter (Misztal1 et al. 2012), Creating a list of related words from keywords and then using recurrent neural networks (RNN) to find a fluent path through finite state acceptors for sentence generation (Ghazvininejad et al. 2016). Other corpus-based poetry generation techniques include using two corpuses, one for grammatical and rhythmic structure and other solely for text mining. (Toivanen et al. 2012).

Poetry Analysis

Genuine poetry can communicate before it is understood.

– T.S. Eliot

What makes this particular poetry more compelling than the others? This part of the review tries to answer this question. Computational Poetry is a research area that is born at the crossroads of research in Computational Linguistics and Artificial Intelligence. Many have tried to solve the problem of machine-generated poetry, however, as for developing an empirical model for describing poetry, efforts have been really less. Before we review the different suggested methods for poetry analysis, I think it's better to redefine poetry relevant only to the discussion of this review.

Restricted Definition of Poetry:

Poetry is just the evidence of life. If your life is burning well, poetry is just the ash.

— Leonard Cohen

This quote justifies how vast, subjective and how difficult it is to define poetry. For our discussion, we'll define poetry as a piece of natural language text that complies with the constraints of grammaticality, meaningfulness, and poeticness.

1. **Grammaticality:** This states that poem should be syntactically well formed. This doesn't necessarily mean that poem should comply with all the grammatical rules in English language, rather poem should be governed by set of poetic grammar rules.
2. **Meaningfulness:** Poem should convey some thought or concept. For instance, a given text may be correct by rhyme and meter to be considered a good poem but if it fails to convey particular emotion or message, it should not be considered as a good poem.
3. **Poeticness:** Poem should follow certain guidelines for style, rhyme and word-stress to be considered as a good poetic text.

Previous work in Poetry Analysis:

As mentioned above very little work has been done in studying empirical models for poetry. In this section, we'll examine a few ways of distinguishing good poetry from average poetry.

Rhyme, Meter and Word Stress. In Automatic Analysis of Rhythmic Poetry [Greene et al. 2010], authors suggest that all the lines in the corpus to be converted into a stress – syllable mapping using an FST. Next to cascade that FST into norm FST which near-iambic parameters to strict iambic parameters. Next step is to use Viterbi's decoding to recover the highest probability alignments.

Here, one of the most important difficulty is poets tend to use a particular word in the beginning or only the endings, few syllables are mostly stressed and a few of them are mostly

unstressed. Therefore, the solution is to apply unsupervised learning to acquire word-stress patterns directly from raw poetry, without relying on a dictionary.

shall i compare thee to a summers day

| | /\ | | | /\ |

S S* S S* S S* S S* S S*

Output of the FST [Automatic Analysis of Rhythmic Poetry [Greene et al. 2010]]

Computational Aesthetics. Up till now, we've been discussing usage the of word-stress, rhyme, and meter to analyze poetry. Though they can be really effective in distinguishing between the nonpoetic and poetic text based on a little of poeticness and grammaticality, from our re-definition of poetry, it doesn't say anything about meaningfulness. One way to analyze poems based on meaningfulness is to use "aesthetic measure" proposed by mathematician G.D. Birkhoff, who formalized beauty as a ratio between order and complexity (Birkhoff, 1933). To analyze poetry based on aesthetics, various elements of the art that help human analyze the poem needs to be studied. In their computational analysis, Justine Kao and Dan Jurafsky chose to study – diction, phonetics, affect, and imagery.

Diction: To understand the poetry flow from the diction point of view several arguments regarding good poetry writing should be considered. Few such arguments include - Good writing consists of a balance of ordinary words that make the writing comprehensible and strange words that make the writing distinguished (Aristotle, 1998), Poetic language is usually intentionally ambiguous and often packs several meanings into a compact passage (Addonizio & Laux, 1997)

et al. Hence, two probable methods to study diction could be word-frequency count – measure of difficult words and Type-Token Ratio – the measure of diversity.

Phonetics: Rhyme and sounding are critical in distinguishing a good poem from a fairly average one. The way the poem sounds impacts its acceptance. Few ways to determine a good sounding poem would be studying: perfect and end slant rhymes, Alliteration and Consonance, Assonance et al.

Rhymes is the most popular and well-used sound device to drive the poetry further. One way to analyze rhyme would be examining phoneme sequences at the end of the line. Identical phenome sequences from the stressed vowel indicate a perfect rhyme structure. Alliteration is the repetition of the consonant sound at the beginning of words and consonance is the repetition elsewhere. Alliteration is calculated by checking if the initial phoneme of two consecutive words is identical. Consonance is calculated by checking if there are at least two identical phonemes in the given window. Assonance is calculated same as consonance but it is the repetition of the vowel sound.

Example - Perfect end rhyme: floor / store, Slant end rhyme: bred / end , Alliteration: frozen field, Consonance: brown skin hung, Assonance: shallower and yellowed.

Affect: One of the most widely used ways to determine the affect of the poetry is to use the sentiment analysis.

Example - Positive outlook able; friend, Negative outlook abandon; enemy, Positive emotion happiness; love, Negative emotion fury; sorrow, Physical wellbeing alive; eat, Psychological wellbeing calm; adjust .

Imagery: Good writing is about showing and describing the scene rather than just telling the situation. One way to collect statistics regarding imagery would be counting down how many times did the writer used concrete object instead of abstraction or generalization.

Example - Object boat; leaf, Abstract day; love, Generalization none; all

Poetry Generation

Will technology ever be able to replicate an artist's creativity? This is one of the most frequently asked questions at the dawn of the age of Artificial Intelligence. Many people have tried answering this question by using AI to generate various artistic creations like, Brain.fm[AI generated music startup], Google's Deep Dream project, Google's art machine composing it's own tunes, Sunspring [AI generated movie screen play] and projects like ALFRED [AI based poetry generators]. This section of the review deals with natural language generation, particularly poem generation. Here, we will examine various approaches for poetry generation.

Poetry Generation from POS tags

Most of the earliest poem generators used part-of-speech (POS) tags, Context Free Grammar and Markov Chains to generate poetry emphasizing a little on the three criteria of poetry mentioned earlier.

The basic technique of a POS tag based poetry generation is to extract the POS-tag sequences from the corpus and calculated the probability of each sequence in the verse. Next step is the generation of poetry from this. In POS-tag based poetry generation with WordNet[Agirrezabal et al. 2013], authors describe three way of generating poetry from the probability table.

The first step of the generation part is to create strophes. The concept here is to find the most commonly used patterns in the corpus. The idea is to use a morphological analyzer to extract the POS patterns in the strophes and using the same patterns to build new ones. Next, based on the strophe - we can replace the words according to POS-tags and suffixes and create new strophes, replace nouns and adjectives with equally inflected words to create new clauses or replace only the nouns in the strophes with semantically related noun – for instance, hypernyms or synonyms, to create new strophes. the nouns in the strophes with semantically related noun – for instance hypernyms or synonyms, to create new strophes.

Keyword extraction Poetry Generation

These approaches for poetry generation extracts keywords from the text based on sentiment analysis results and uses those keywords to generate poetry using the poetry based on the structure either given by the user or learned from the corpus.

One such method described in Corpus-Based Generation of Content and Form in Poetry [Toivanen et al. 2012], uses two different corpuses. One corpus provides with the semantic content of the poem while another corpus provides with the form – grammatical and syntactical structure of the poetry. The basic idea here is to use word association network derived from the content corpus. Word association network is built using the word pairs extracted from their frequency co-occurrence in the corpus. Word association network is built using log-likelihood ratio. The network is built as a multinomial network with every pair of words having four probabilities regarding their contingency. This part takes care of meaningfulness aspect of our redefinition of poetry. Next, the grammatical syntax and morphology of the poem are obtained from the grammar corpus. Here the form of a concrete poem is copied and then just it words are changed. This takes care of grammaticality aspect of our definition. Now, the only thing

remaining is to ensure the poeticness of out poem. Poeticness can be ensured using word-stress and stress-syllables mapping in the corpus.

Another of such a method is to extract keywords from the corpus describing particular emotions and then using those keywords to generate poetry in a blackboard based system. The basic idea of the approach is to have different expert agents – for instance, Analysing expert, word generating expert et al. All these experts communicate with blackboard and their results are stored in a centralized pool of ideas. The completing expert will use the pool of ideas to generate the final solution of the system.

Blackboard is a common workplace with partial solutions given by particular experts. The input text, which is a corpus is given to the blackboard. Experts analyze it to create the main theme for the sentimental content for the poem. Next, keyword extracting experts extracts the keywords from the corpus which are related to the required sentiment.

The most important expert in the system is the Emotion Expert. It defines the emotional state of the poem determined by the emotional state of the input text. This emotional state is used in generating every new line for the poem since the emotional state of the entire poem after the line generation should remain same. Sentiments are calculated in terms of valence and arousal. Valence of the text describes positive and negative sentiment strength in the poem. Valence is given by:

$$V = \alpha_{opt} \cdot \sum_{s \in Text} Sent_{pos} + (2 - \alpha_{opt}) \cdot \sum_{s \in Text} Sent_{neg}$$

Similarly, arousal value is given by measuring the average count of arousal symbols like “!”, “!!” et al. occurring in the text. Final emotion of the poem is given by

$emotion = \arg\min d((v_t, a_t),(v_x, a_x))$, where $x \in S$

Next, the words generating experts have the lexical knowledge and they generate antonyms, synonyms, hypernyms et al. for the keywords to be used for poetry generation. Apart from these, many other experts like metaphor expert, rhetorical experts et al. can be added to the blackboard system to generate metaphors etc. for a few keyword phrases in the system. The poems produced by this system, in general, satisfies all the three constraints of poetry generation.

Poetry Generation as a state space search problem

Poetry generation can be viewed as a state space search problem, where a state in the search space is a possible text with all its underlying representation, from semantics all the way down to phonetics. By viewing poetry generation as the Stochastic Search problem, the state which satisfies the given target structure of the poetry the most is the required state. Various searching algorithms like genetic algorithms, hill climbing or heuristic based searching algorithms like A* can be applied to generate poetry.

Using Hill Climbing. Model poetry generation can be viewed explicitly as a searching problem. In, Towards a Computational Model of Poetry Generation [Manurang et al. 2000], authors address hill climbing approach to poetry generation. The basic idea is to generate a phrase and evaluating the phrase based on given evaluating parameters like for instance, Phonetics, Grammar, and Syntax et al. to ensure the three qualities of the poem.

Next step is to evolve the phrases by adding new phrases. Here, the candidate pool with the best source is generated and for every candidate in the pool, the search process is repeated until the most qualifying candidate is found. Three basic operators for evolving a phrase are:

1.) **Add:** “He walked” => “He walked to the beach”

2.) **Delete:** “He likes to drink tea and run” => “He likes to drink tea”

3.) **Change:** "He *ran a fast as he could* to catch the bus " => "He *sprinted* to catch the bus"

The only problem with hill climbing search is our, search may reach the local maxima, which could be *nearly* as perfect as the target poetry but we will never know if it is the best solution to our search problem.

Using Genetic Algorithms. To tackle the problem of local maxima encountered in the Hill Climbing search, another workaround could probably be using evolutionary algorithms to generate poetry. Here also, the problem is viewed as a stochastic search problem. The basic difference here is the usage of Lexicalized Tree-adjoining grammar or LTAGs as the data structure to ensure grammaticality of the solution as well as to mutate the candidates.

In LTAGs, grammars are based on the composition of elementary trees or auxiliary trees which define the most minimalistic linguistic structures. At every mutation, these trees are added to a derivation tree or deleted depending if they meet the linguistic structure or not. As shown in Figure 1 (a), all the words have elementary lexical tree derived from the corpus and derived trees are formed using add-subtrees, delete-subtrees and swap-subtrees operators. These operators randomly add, deletes and swaps subtrees from derivation tree until a candidate with syntactically correct derivation tree is found. This is used for mutating parents.

Here, poeticness is the measure of meter and rhyme in the generated text. Here, poeticness is evaluated simply by maximizing the similarity with the target form. For instance, target form for haiku would be [x,x,x,x,x,b, x,x,x,x,x,x,x,c, x,x,x,x,x,b] where be denotes the rhyming syllable and x denotes random syllable that makes sense in the context.

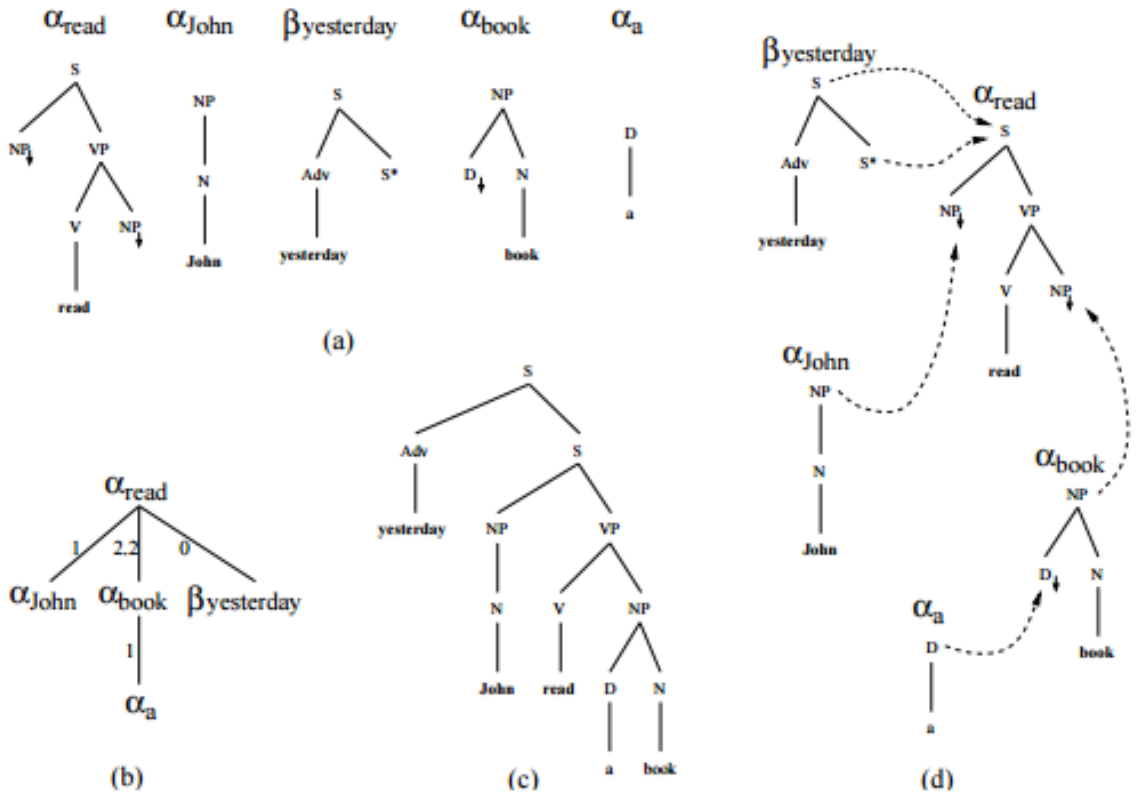

Figure 1 LTAGs with different genetic operators. Using Genetic Algorithms to Create Meaningful Poetic Text [Manurang et al. 2012]

As for meaningfulness, it is similar to poeticness over here. Meaningfulness evaluating function simply tries to maximize the similarity between the target semantics, a specification of the meaning an optimal solution should convey, and a candidate semantics, the meaning conveyed by a candidate solution. Here similarity is measured by comparing two structures based on structural and conceptual similarities. The basic idea for comparison is that two structures are similar if and only if they share a common literal functor. The only drawback here is that the similarity mapping between two forms is an NP-Hard Graph isomorphism problem which is very expensive computationally compared to hill climbing.

Neural Networks for Poetry Generation

One of the most recent techniques suggested for poetry generation is the usage of Neural Networks and Deep learning for poetry generation. One advantage of using a deep learning/neural network based language generator is the possibility of using distributed representation of words with a single parameter describing a huge range of target semantics. Apart from this, another important advantage of using deep learning and neural networks is the possibility of hierarchical modeling, which in turn can create sentences from characters based on constraints.

Generating Tropical Poetry using RNN. As seen Neural Networks language generation are less expensive than viewing language generation as a state search problem as described in the previous section. One of the approaches, would be to use a computing stress-syllable patterns for a large dataset of dictionary words. Which is then used create a list of related words for the topic for poetry.

Next, to ensure meter and rhyme, rhyming word pairs for the endings of our lines, are selected from related word list based on predefined rhyming structure, which for example could be something like *ABAB CDCD EFEF GG*.

Next step would be to create a huge Finite State Acceptor, with a path for every conceivable sequence of words that meet our constraints is created. The FSA should compactly encode all the rhyming word sequences that also obey our formal constraints. These constraints are enforced by encoding line number and syllable count in the FSA paths. For instance, “L1-S5” would mean that our operator is in Line 1 and it has seen 5 syllables so far.

Next step would be selecting a fluent path through the FSA using the RNN. Here, we train the two layered RNN with our corpus and a Long Short Term Memory. Next step is to use beam search to decode the language model guided by the FSA. This helps to keep poem on the topic and ensure meaningfulness apart from poeticness and grammaticality (which are ensured by FSA and trained RNN)

Experimental Results

For poetry generation, I chose to implement highly expensive genetic algorithm technique as well as very naïve n-gram based POS tagging based system.

POS tagging based system This is based on early naïve poetry generation models based on POS and Markov chains. First, POS tags are used to tag the entire corpus. Next, Hidden Markov Model is used to analyze each POS and it is iterated over to create strophes. Which are then merged to give the output.

head in this burned out paradise tonight there 's getting not here much too

needs is those that know it thinking that the pages am i 'm going

own there made exchanging all down your own there was really want and things

anyway you out here you used going to end my warnings there got new

and seems all down your own choice

there were once was not to all down my twisted but baby for more

nothing to lend your present now hard when he who is home here standing

out here me if in here them that 's my minds created out here

Example output poem of the system

Genetic Algorithm based state space search this implementation is inspired from - Using Genetic Algorithms to Create Meaningful Poetic Text [Manurang et al. 2012]. CMU dictionary and ltagextract libraries were used for rhymes and LTAG extraction.

Will later be past

Before she slips in the sand

Will later be fast

Example Haiku Created by the System

Analysis and Comparison of both Methods It can clearly be seen that the second algorithm has a proper structure, rhyme, and meaningfulness compared to POS and Markov Chains based implementation. This is largely due to enforcing our definition on the second implementation as well as the fact that the Hidden Markov Model doesn't actually force the structure since it just helps in creating strophes which may or may not satisfy the structure. Poeticness is given importance in the second implementation while there is no rhyme and meter per say in the first implementation hence it looks more like word salad. In genetic algorithm-based technique, LTAGs does a great job in creating syntactically proper lines while POS tagging and Markov Scores, though does a good job in creating meanings in parts of the line, it does a terrible job in creating meaning and structure for the whole verse.

Conclusion

In conclusion, I think, it's safe to say that Poetry Analysis and Generation is a lot different that Natural Language generation and simple text analysis due to poetry's unity. This is mainly due to the fact that a good poem is a mixture of Lexis, Rhymes, Meters, Syntax et al.

In this literature review, I've tried to examine and explain various poetry analysis and poetry generation technique. I've also compared between various solutions based on simple metric of evaluating poetry based on grammaticality, meaningfulness, and poeticness. I would like to argue that although a lot of work is done on poetry generation, a little is done on poetry analysis and more work on poetry analysis would help make better algorithms for generating such texts.

This review also compares between various methods for poetry generation. Although there has been a huge improvement since early attempts like ALFRED to the current elegant solution using RNNs. Towards the end, I've tried to compare and contrast Markov chain based and state space search based techniques where state search based technique comes out as a clear winner.

References

[1] Automatic Analysis of Rhythmic Poetry with Applications to Generation and Translation [Greene et al. 2012]

[2] Towards A Computational Model Of Poetry Generation [Manurung, et. al 2000]

[3] Using Genetic Algorithms to Create Meaningful Poetic Text [Manurung, et. al 2009]

[4] Generating Topical Poetry [Ghazvininejad et al. 2016]

[5] Poetry generation system with an emotional personality [Misztal et al. 2014]

[6] Corpus-Based Generation of Content and Form in Poetry [Toivanen et al. 2012]

[7] POS-tag based poetry generation with WordNet [Agirrezabal et al. 2013]

[8] Computing Poetry Style [Delmonte 2000]

[9] A Computational Analysis of Style, Affect, and Imagery in Contemporary Poetry [Kao & Jurafsky 2012]