jcls. Journal of Computational Literary Studies

OPEN ACCESS

Article

Limericks and Computational Poetics: The Minimal Pairs Framework

Computational Challenges for Poetic Analysis and Synthesis

Almas Abdibayev[1]
Yohei Igarashi[2]
Allen Riddell[3]
Daniel Rockmore[1]

1. Department of Computer Science, Dartmouth College, Hanover, USA.
2. Department of English, University of Connecticut, Storrs, USA.
3. Department of Information and Library Science, Indiana University, Bloomington, USA.

Citation
Almas Abdibayev, Yohei Igarashi, Allen Riddell, and Daniel Rockmore (2022). "Limericks and Computational Poetics: The Minimal Pairs Framework". In: *Journal of Computational Literary Studies* 1 (1). 10.48694/jcls.117

Date published 2022-12-07
Date accepted 2022-03-30
Date received 2021-12-23

Keywords
limericks, computational poetics, minimal pairs, evaluation, language models

Reviewers
Natalie Houston
Ted Underwood

Note
This paper has passed through the conference track of JCLS. In addition to being peer reviewed, it was presented and discussed at the 1st Annual Conference of Computational Literary Studies 2022 at the Technical University of Darmstadt.

Abstract. Computational poetics encompasses the wide range of challenges implicit in analyzing and generating poetry – in all of its many forms – through computational techniques and frameworks. In this paper, we build on a nascent body of work that has proposed the use of the limerick as a 'model organism' for computational poetics, and in particular the use of Benchmarked Poetic Minimal Pairs (BPoMP) as an investigative framework, especially for the evaluation of the poetic abilities of deep learning language models. To that end, we include results for two new BPoMP tasks of interest for limerick analysis – the word deletion task and the limerick completion tasks. We include a release of a data set for the deletion task. We also offer up a suite of an additional ten BPoMP challenges whose precise formulations still require detail.

1. Introduction

> Much less would I care to try sliding [limericks] through the ... apertures of a calculating machine, in order to discover the leading "traits" or themes with which they are concerned, even assuming that anything meaningful could be learned in such a way. – Gershon Legman, *The Limerick* (1969)

What do computers 'know' or recognize about poetic form? And can they 'learn' about poetry? This paper explores such questions under the heading of 'computational poetics', using limericks as a paradigmatic case or a model organism, first introduced in Abdibayev et al. (2021a) and elaborated on below. We use an experimental framework called 'minimal pairs' to examine the extent to which language models (Jurafsky and Martin 2021) can discern elements of poetic language and form as well as the poem's overall integrity.

Nearly fifty years ago, the folklorist and limerick historian, Gershon Legman, expressed his distaste at the very idea of the computational analysis of limericks, a particularly folk poetic kind (see epigraph). But despite these admonitions, recent work (Abdibayev et al. 2021a,b) has focused attention on the limerick for several reasons, viewing the

limerick as – borrowing from the life sciences – a 'model organism' for computational poetics.

In the life sciences, many disciplines rely on model organisms: A handful of organisms are studied for their 'representational scope', that is, their ability to stand in for many other organisms and phenomena and thereby "create knowledge that can be projected beyond the immediate domain in which it was produced". For example, the weed known as thale cress is a key model organism for the broader study of the genetics, evolution, and development of many plant species (Ankeny and Leonelli 2020). Other familiar model organisms include the fruitfly (drosophila), the roundworm (the nematode *C. elegans*), and the mouse (*Mus musculus*). Each has the property of simplicity – at least relative to their larger research environment – as well as some degree of pliability and clarity *vis-a-vis* interrogative pathways. That is to say, model organisms are generally chosen both for the ease with which a potentially influential parameter can be isolated and then tweaked as well as the ability to understand the effect of that modulation on a phenomenon of interest. *C. elegans* has only 1,000-3,000 cells (depending on how you count), a few hundred neurons, and about 20,000 genes. Drosophila turn over a generation every week. Questions of evolution, genetic engineering, and neuroscience have the potential of being answered at these scales of time, space, and components, and with that, provide a solid platform for broader speculation. In general, model organisms have been critical for the important advances that have been made over at least the past half-century (including several Nobel Prizes) in human genetics, neuroscience, reproductive science, botany, and biology.

Poetry – the complex interplay of sounds, rhythm, words, meanings, narrative, visual formatting, and more – is manifested across a wide range of forms. Such diversity can prove challenging in the search for general principles that might apply across computational approaches. Hence, isolating a particular form like the limerick provides a good place to start. The limerick is a relatively short and simple form that happens to have a high density of poetic features: five verse lines with an *aabba* rhyme scheme and a 3-3-2-2-3 accentual-metrical arrangement; the presence of trisyllabic feet, i.e., anapests, dactyls, amphibrachs, depending on how one recites or hears the poem; and usually a condensed, humorous narrative structure (for a fuller discussion of the limerick form, see Preminger et al. (1993)). These features can be manipulated, as we have done in our various experiments to date. Limericks also serve as a valuable model because language models in widespread use today tend to require short texts, and the limerick form has high linguistic-formal interest relative to its brevity (Liu et al. 2019). The notion of a model organism for literary study was first popularized in the seminal paper of Mary Poovey Poovey (2001), who argued that lyric poetry served as the model organism for literary criticism. We hew somewhat more closely to the analogy and inspiration from the sciences. The limerick form is an experimental and analytic environment where progress is highly likely, and our method and findings may be generalizable to other short poetic forms (for example, epigrams, haiku, clerihews, quintains, and even sonnets) – and, beyond that, to longer poetic forms and potentially literary language generally.

We therefore join existing work in computational approaches to poems in English, both those engaged in machine reading and machine writing. We contribute to work that

seeks to automate the detection and analysis of poetic features, language, or kinds (for example, see Anttila and Heuser (2016), Houston (2014), and Long and So (2016)) and work where computers are trained to output or compose poetry (for example, see Ghazvininejad et al. (2017) and Lau et al. (2018)). In particular, Long and So's work in 'literary pattern recognition' and their stylistic taxonomy of the haiku inform our work with a similarly short poetic form. More generally, we also take our cue from foundational applications of machine learning to literary texts (Bode 2018; Long 2021; Moretti 2017; Piper 2018; So 2020; Underwood 2019). The model organism of the limerick also promises to contribute to formalist investigations of English poetry. Our project complements work like the Princeton Prosody Archive and other endeavors that, in concert with the 'New Formalism' and 'Historical Poetics', have brought sustained attention to poetic form in literary study in recent years.[1] Finally, we build here on other recent work in computational poetics and language modeling: the minimal pairs method was introduced for limericks (Abdibayev et al. 2021b) and then slightly expanded in Abdibayev et al. (2021a) with a system for detecting some of the main features of the limerick form while also producing a publicly available data set of limericks.[2] We make use of that data set herein.

Our main contribution in this paper is to continue the expansion of the testbed of 'minimal pairs' challenges for poetry, the "benchmark of poetic minimal pairs" (BPoMP) (Abdibayev et al. 2021b), which are inspired by the "benchmark of linguistic minimal pairs" (BLiMP) framework (Warstadt et al. 2020). We describe BLiMP and BPoMP in greater detail below. The first poetic minimal pair tests evaluated the degree to which language models could detect limerick end rhymes from non-rhymes and the overall limerick structure (Abdibayev et al. 2021b). In this paper, we report on a test set and results pertaining to new BPoMP challenges: word deletion and a synthetic fifth line. The former tests if a language model can distinguish a given limerick from a version of it with missing words (in the sense of identifying the former as more limerick-like). The latter creates the challenge of distinguishing an original limerick from a version of it where the original fifth line has been replaced by a computer-generated one. Both of these challenges recall various aspects of literary and textual practice, from erasure poetry to popular limerick completion contests held during the early twentieth-century "great limerick boom" (McInerney 2001).

We release a new BPoMP data set concurrently with this paper, freely and publicly available, thereby enabling reproducibility of results in computational poetics. The combination of a curated and publicly available corpus of material with open source models produces a "standard package" (in the sense of Fujimura (1992)) for deep learning in computational poetics, and creates new opportunities for other computational literary studies scholars to engage with the machine learning techniques and tools for critical and creative work in poetry and literature. This research enhances literary scholarship by providing a testbed for evaluating the extent to which computers can analyze and compose short verse. Furthermore, a set of 'benchmarked' computational poetic tasks

1. See, e.g., The Princeton Prosody Archive (https://prosody.princeton.edu/) and the essays deriving from it.
2. The collection of limericks used therein is available at Zenodo: https://zenodo.org/record/5722527. These limericks comprise a cleaned subset of a larger corpus, also filtered as best as possible to adhere to formal limerick structure as well as to exclude offensive language. See the documentation at the site as well as the paper referenced in text.

creates a familiar setting for computer scientists and especially the deep learning community by articulating measurable targets for interrogating the poetic capabilities of current and future language models. In addition to the deep exploration of deletion and completion tests explored below, we include a suite of new tests, whose design – which can be subtle – is still underway. We hope that by introducing this next set of tests herein we are able to foster interest and collaboration in the broader community in the BPoMP schema. In the next section we give some more background on language models, BPoMP in general, and our two new BPoMP challenges. In section 3, we explain in detail the deletion tests and the completion test, and our results. The section 4 is a discussion of the tests, including implications for poetics. We close in section 5 with discussions of future work.

2. Background

In this section, we give a brief overview of the language models that we are evaluating using our minimal pairs method. We then give more detail on the BPoMP construct as well as some discussion of the word deletion and last line completion minimal pairs. We also describe the corpus (OEDILF) from which we source our limerick data set and the filtering process that produces the data sets for these experiments.

2.1 Language Models

The renaissance of neural network models (often marketed under the heading of 'deep learning') has greatly advanced expectations for a machine's ability to perform machine reading and machine writing. Applied to language modeling (Goldberg 2017; Jurafsky and Martin 2021), these models present exciting opportunities for research in literary studies and computer-supported creative work.

Our work explores the power of the GPT-2, BERT, TransformerXL, and (causal) XLNet language models. Each is based on the computationally efficient 'Transformer' architecture (Vaswani et al. 2017), a basic mathematical model that, given some text, predicts with varying probabilities the surrounding text in any human-produced text instance. This is an encoding of each word in the vocabulary as a *vector*, which is effectively a list of numbers. This model depends on a range of numbers – parameters – that have been set according to the likelihoods of various text strings occurring in a large body of text, such as all the writing in Wikipedia. Pre-training is the algorithmic setting of these parameters based on the example text corpora.

The Transformer architecture derives from the better known ideas and architectures based on and inspired by "neural networks" (see e.g., Gurney (1997)), which are themselves loosely modeled on the 'wet' network of neurons in our own brains: connected collections of billions of simple cells (neurons) whose signaling patterns underlie the abilities of all animals to encode learning and learned behaviors. Early neural networks with a relatively small number of (mathematical) neurons were but one of a large number of basic 'classifiers', mathematical models for segmenting data and predictive

algorithms.[3] It was something of a surprise that when the model sizes were dramatically increased – going from tens of parameters to orders of magnitude larger – that neural networks showed great and in many ways unforeseen abilities to do data discrimination and prediction. Modern machine learning continues to ride the wave of the strength of these architectures and modern computing has enabled the ability to continuously fit more and more parameters in models of increasing size and complexity.

'GPT' stands for Generative Pre-trained Transformer. GPT, GPT-2, and now GPT-3 (Brown et al. 2020; Radford et al. 2019) are the three generations of a basic – GPT – architecture specially designed to produce human-level text. They come in 'sizes' (small, medium, large, etc.) and the successive generations are largely distinguished by the expansion of the number of parameters embedded in the models and the requisite sizes of the (pre-)training sets needed to tune these models – tens of billions in the case in GPT-2 and hundreds of billions in the case of GPT-3.

'BERT' stands for Bidirectional Encoder Representations from Transformer. The 'bidirectional' modifier reflects a model that looks at word sequences both backwards and forwards for training (Devlin et al. 2018).

'TransformerXL' (Dai et al. 2019) is a causal language model, meaning that unlike BERT it only uses preceding words to predict the next word. Its standout ability is the addition of a recurrence mechanism, which is a form of machine memory. Information from previous word sequence segments processed by the model (a fixed-length context window of words that the model uses to predict the next word) is carried over to the next segment, which theoretically allows TransformerXL to look farther behind in text to make a good prediction.

'XLNet' is an evolution of aforementioned TransformerXL model that uses a clever mathematical trick to approximate all possible orders of words in a sentence, where, instead of a fixed-order (left to right) context, the model is exposed to a randomly permuted sequence of all words that both precede and succeed the word we are predicting, while predicting the last word (or last few words) of this sequence. Since the new context includes tokens both from the left and right of the original context of the target word, the model is bidirectional like BERT. At the same time it can also be used for left-to-right decoding (i.e., generation), like GPT-2 (Yang et al. 2019).

2.2 BPoMP

The BPoMP method evaluates a language model by giving it a choice between a right and wrong instance, where the wrong instance is a 'minimally' doctored version of the correct – original – instance. For example, one might have *Poem A*, but then replace a single word in *Poem A* with a random word to create *Poem B*. The more capacious a language model, the more reliably it is able to distinguish between the original and the corruption. The design of minimal pairs to isolate a phenomenon of interest can be rather subtle.

3. The origin story goes back to McCulloch and Pitts's original modeling of neural activity (McCulloch and Pitts 1958) and later, Rosenblatt's invention of the 'perceptron', a simple mathematical model of a neuron (Rosenblatt 1943).

There are three reasons to use BPoMP challenges when comparing language models' ability to model poetry. First, the BPoMP challenges provide a useful 'second opinion' when considering model performance. Traditional measures of model 'fit' such as perplexity (see e.g., Chen et al. (1998)) are often unreliable or difficult to calculate in settings where the observed data is high dimensional. Performance on BPoMP challenges, by contrast, is easy to calculate and to interpret. Second, the BPoMP challenges can be used in settings where traditional evaluations such as perplexity are unavailable. As examples of this, BPoMP can be used to compare two language models which use different tokenization strategies and to compare a bidirectional language model with a traditional ('causal') language model. Third, using the BPoMP challenges to evaluate models can yield insights into the strengths and weaknesses of particular models. It is, for example, easy to imagine one language model performing better on BPoMP challenges involving rhyme but worse on all other challenges. Such a result may indicate that some component of the model is doing well at capturing regularities in language that relate to rhyme. If this component can be isolated, it could be borrowed by other models. While one might balk at leaving such evaluations of models to the machine, recent work supports the use of machine – rather than human – evaluation (Clark et al. 2021).

The preceding arguments in favor of the BPoMP challenges closely resemble those offered in favor of the BLiMP challenge set (Warstadt et al. 2020). Whereas BLiMP helps to bring into focus the strengths and weaknesses of language models' ability to adequately model differences between grammatical and ungrammatical sentences, BPoMP helps researchers to characterize models' capacities to capture differences between poetic and non-poetic language.

In Abdibayev et al. (2021b) a dataset of 10,000 minimal pairs of limerick/corrupted limericks was used in a task of 'choosing' between the two options to determine the original limerick. Transformer-based models were used. The minimal corruptions were: (1) shuffling two rhyming end-of-the-line words, (2) shuffling two rhyming lines, (3) replacing an end-of-the-line word by a non-rhyming synonym. While the models identified the original limerick at rates better than chance, there was a good deal of room for improvement. It is fair to say that the models have yet to demonstrate that they have developed an ear for poetry.

This work on detecting formal elements (poetic analysis) complements, but takes a fundamentally different approach from, useful, existing prosodic parsers and pedagogical tools – e.g., *Prosodic* and *For Better For Verse*.[4] Whereas such tools are focused on discerning or teaching meter, our standardized package is concerned with investigating more fundamentally – via our adaptation of the minimal pairs method – what 'poetic knowledge' a language model possesses or can learn, and therefore has a more comprehensive scope which includes not only accentual patterning but other formal elements (rhyme, musical devices like alliteration, and so on).

4. On *Prosodic*, see Porter (2018). The longstanding digital humanities tool *For Better For Verse* can be found at https://scholarslab.lib.virginia.edu/work/for-better-for-verse/.

2.3 Word Deletion

Word deletion is the focus of one of the new minimal pair benchmarks in this paper. In short, a language model is presented with a limerick and a near twin, corrupted by the removal of one, two, or three words. The model then 'chooses' between the two by calculating the likelihood of both poems. The text with the higher likelihood wins. This textual challenge probes the models' ability to discern the semantic, syntactical, and grammatical integrity and form of the limerick genre. Word deletion should disturb these qualities and a language model should know this in the probabilistic sense in which it 'knows'.

The word deletion tasks also recalls certain textual and literary practices. Presenting our models with an original text and a version of the same text with missing words mimics the longstanding problem of omissions in the historical transmission of texts. Missing words are an inevitability in the manuscript documentary record, and the scholarly practice of textual criticism has codified various kinds of deletions. For example, a manuscript might be missing words because of *saut du même au même*: When the same word is repeated in close proximity, the scribe "copies the text as far as its first occurrence ... then looking back at the exemplar to see what he must copy next he inadvertently fixes his eye on the second occurrence of the word and proceeds from that point". The result is that "the intervening words are omitted from his copy" (Reynolds and Wilson 1991). Missing words and other such common errors degrade the transcription, but are crucial for textual scholars in positing the relationships between different manuscripts (that is, the practice of stemmatics or stemmatology): Missing words can help to establish how related or unrelated a given manuscript is to other manuscript copies of the 'original' (the archetype) (Reynolds and Wilson 1991). The deletion-based BPoMP arguably resembles the first and most fundamental step in such textual scholarship – to identify the more 'correct' or more likely text, which is not missing words – and measures how well models can automatically carry out this task.

In a different literary context, the minimal pairs challenge also evokes erasure poetry. An erasure poem is a type of found poem, where an existing composition written by another is manipulated to create a new work through the blacking out or omission of some words or letters (American Poets 2022). Another way to look at the deletion task, then, is that our algorithm is taking an original limerick and creating an erasure poem out of it. This task raises all sorts of questions about the human and machine discernment of erasure poems. What exactly might distinguish 'good' erasure poetry from the randomized omission of a composition's original words? Although we are assuming for our purposes that the erasure poem is the 'incorrect' choice, how might models learn to recognize legitimate or artistically compelling instances of erasure poems? The task also raises interesting ontological questions about the nature of found and manipulated poetry like erasure poems.

2.4 (Poor) Last Line Completion

Our second minimal pair experiment tests models on their ability to distinguish an existing limerick from a similar limerick in which the fifth line of the original limerick has been overwritten by a computer-generated fifth line. The first four lines of each

poem in the pair are identical. This experiment involves, then, a stage where a natural language generation technique – whatever technique one uses – completes a four-line fragment of a limerick with a plausible fifth line that fulfills some of the requirements of the limerick form: primarily the *a* end rhyme (that rhymes with lines 1 and 2 in the *aabba* rhyme scheme) and a line length that also generally conforms to those of lines 1 and 2 (typically the longer lines of the limerick in terms of the number of words, compared to lines 3 and 4). (In our experiment, we did not require the synthetic fifth line to meet the rhythm requirement of three stresses and triplet meter.)

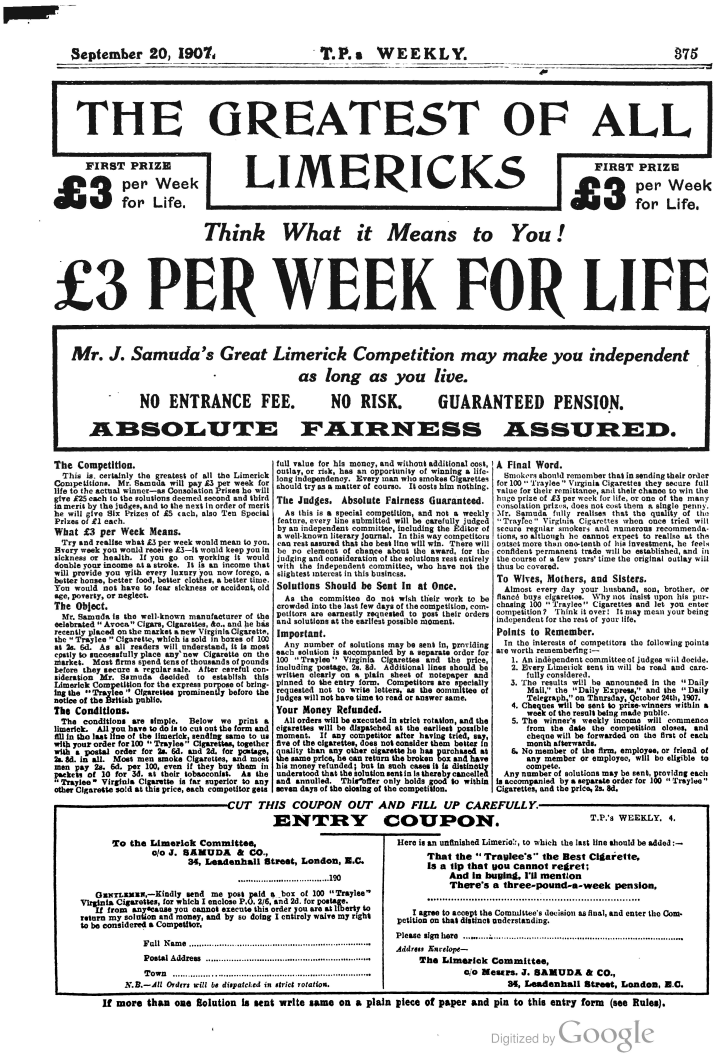
This challenge asks the language model being evaluated to discern the original limerick from the synthetic version. This is testing whether or not the model has some minimal sense of how limericks typically end. The machine-generated last line is not a typical line, but it is a good fake, so a model purporting to model narrative and/or semantics cannot 'cheat' by just looking to see if it has the right number of syllables and rhyme scheme. Thus, the last line completion minimal pairs also probe a language model's ability to encode coherence in the limerick. Note that is possible that a language model performs well on word deletion and not on line completion or vice versa.

This test pushes the definition of a 'minimal' alteration to its limit in that we replace an entire line of a limerick. Yet this aspect of the experiment recalls the history of the limerick form. Although the origins of the limerick form are obscure, we do know that it was initially popularized in the nineteenth century by Edward Lear's *A Book of Nonsense* (1846) and then reached another peak of popularity around the turn of the twentieth century. During this latter peak – called by one limerick historian "the great limerick boom" – several promotional competitions elicited the public to submit a final line for a limerick fragment, for a chance at winning substantial prize money (McInerney 2001). The most well-documented example in the limerick literature is a contest put on in 1907 by the cigarette company *J. Samuda & Co.*(Legman 1969). Offering a prize of 3£ per week for the rest of the winner's life, Samuda's contest asked the public to complete the following limerick fragment, which advertised the company's product:

> That the "Traylee's" the Best Cigarette,
> Is a tip that you cannot regret:
> And in buying, I'll mention
> There's a three-pound-a-week pension, ...

J. Samuda & Co.'s advertisement proclaimed that the competition would identify "the greatest of all limericks", but really, as the ad itself reveals, the competition was a gimmick aimed at promoting a new product, *Traylee Cigarettes* (see Figure 1). Hence contestants needed to mail in an order for the cigarettes in order to submit a fifth line for the limerick. Evidently, *J. Samuda & Co.* went on to hold several such promotional competitions involving limerick completion, alongside other similar competitions put on by others. In 1907 alone, there were more than 11 million postal orders for such limerick contests. The winning line for *J. Samuda & Co.*'s 1907 contest was, "Two good 'lines' – one you give, one you get", punning on 'line' (the poetic line, the cigarette, and the financial life line) (McInerney 2001).

In any case, what matters here is less the history of limerick contests *per se* than the fascinating resonances between the poetic challenges we have set for language models



September 20, 1907. T.P.'s WEEKLY. 375

THE GREATEST OF ALL LIMERICKS

FIRST PRIZE £3 per Week for Life. FIRST PRIZE £3 per Week for Life.

Think What it Means to You!

£3 PER WEEK FOR LIFE

Mr. J. Samuda's Great Limerick Competition may make you independent as long as you live.

NO ENTRANCE FEE. NO RISK. GUARANTEED PENSION.

ABSOLUTE FAIRNESS ASSURED.

The Competition.

This is, certainly the greatest of all the Limerick Competitions. Mr. Samuda will pay £3 per week for life to the actual winner—as Consolation Prizes he will give £25 each to the solutions deemed second and third in merits by the judges, and to the next in order of merit he will give Six Prizes of £5 each, also Ten Special Prizes of £1 each.

What £3 per Week Means.

Try and realise what £3 per week would mean to you. Every week you would receive £3—it would keep you in sickness or health. If you go on working it would double your income at a stroke. It is an income that will provide you with every luxury you now forego, a better home, better food, better clothes, a better time. You would not have to fear sickness or accident, old age, poverty, or neglect.

The Object.

Mr. Samuda is the well-known manufacturer of the celebrated "Arcon" Cigars, Cigarettes, &c., and he has recently placed on the market a new Virginia Cigarette, the "Traylee" Cigarette, which is sold in boxes of 100 at 2s. 6d. As all readers will understand, it is most costly to successfully place any new Cigarette on the market. Most firms spend tens of thousands of pounds before they secure a regular sale. After careful consideration Mr. Samuda decided to establish this Limerick Competition for the express purpose of bringing the "Traylee" Cigarettes prominently before the notice of the British public.

The Conditions.

The conditions are simple. Below we print a limerick. All you have to do is to cut out the form and fill in the last line of the limerick, sending same to us with your order for 100 "Traylee" Cigarettes, together with a postal order for 2s. 6d. and 2d. for postage, 2s. 8d. in all. Most men smoke Cigarettes, and most men pay 2s. 6d. per 100, even if they buy them in packets of 10 for 3d. at their tobacconist. As the "Traylee" Virginia Cigarette is far superior to any other Cigarette sold at this price, each competitor gets full value for his money, and without additional cost, outlay, or risk, has an opportunity of winning a life-long independency. Every man who smokes Cigarettes should try as a matter of course. It costs him nothing.

The Judges. Absolute Fairness Guaranteed.

As this is a special competition, and not a weekly feature, every line submitted will be carefully judged by an independent committee, including the Editor of a well-known literary journal. In this way competitors can rest assured that the best line will win. There will be no element of chance about the award, for the judging and consideration of the solutions rest entirely with the independent committee, who have not the slightest interest in this business.

Solutions Should be Sent in at Once.

As the committee do not wish their work to be crowded into the last few days of the competition, competitors are earnestly requested to post their orders and solutions at the earliest possible moment.

Important.

Any number of solutions may be sent in, providing each solution is accompanied by a separate order for 100 "Traylee" Virginia Cigarettes and the price, including postage, 2s. 8d. Additional lines should be written clearly on a plain sheet of notepaper and pinned to the entry form. Competitors are specially requested not to write letters, as the committee of judges will not have time to read or answer same.

Your Money Refunded.

All orders will be executed in strict rotation, and the cigarettes will be dispatched at the earliest possible moment. If any competitor after having tried, say, five of the cigarettes, does not consider them better in quality than any other cigarette he has purchased at the same price, he can return the broken box and have his money refunded; but in such cases it is distinctly understood that the solution sent in is thereby cancelled and annulled. This offer only holds good to within seven days of the closing of the competition.

A Final Word.

Smokers should remember that in sending their order for 100 "Traylee" Virginia Cigarettes they secure full value for their remittance, and their chance to win the huge prize of £3 per week for life, or one of the many consolation prizes, does not cost them a single penny. Mr. Samuda fully realises that the quality of the "Traylee" Virginia Cigarettes when once tried will secure regular smokers and numerous recommendations, so although he cannot expect to realise at the outset more than one-tenth of his investment, he feels confident permanent trade will be established, and in the course of a few years' time the original outlay will thus be covered.

To Wives, Mothers, and Sisters.

Almost every day your husband, son, brother, or friend buys cigarettes. Why not insist upon his purchasing 100 "Traylee" Cigarettes and let you enter competition? Think it over! It may mean your being independent for the rest of your life.

Points to Remember.

In the interests of competitors the following points are worth remembering:—

1. An independent committee of judges will decide.
2. Every Limerick sent in will be read and carefully considered.
3. The results will be announced in the "Daily Mail," the "Daily Express," and the "Daily Telegraph," on Thursday, October 24th, 1907.
4. Cheques will be sent to prize-winners within a week of the result being made public.
5. The winner's weekly income will commence from the date the competition closes, and cheque will be forwarded on the first of each month afterwards.
6. No member of the firm, employes, or friend of any member or employee, will be eligible to compete.

Any number of solutions may be sent, providing each is accompanied by a separate order for 100 "Trayleo" Cigarettes, and the price, 2s. 8d.

CUT THIS COUPON OUT AND FILL UP CAREFULLY.

ENTRY COUPON. T.P.'s WEEKLY, 4.

To the Limerick Committee,
c/o J. SAMUDA & CO.,
34, Leadenhall Street, London, E.C.

.....190

GENTLEMEN,—Kindly send me post paid a box of 100 "Trayleo" Virginia Cigarettes, for which I enclose P.O. 2/6, and 2d. for postage.

If from any cause you cannot execute this order you are at liberty to return my solution and money, and by so doing I entirely waive my right to be considered a Competitor.

Full Name .....

Postal Address .....

Town .....

N.B.—All Orders will be dispatched in strict rotation.

Here is an unfinished Limerick, to which the last line should be added:—

That the "Trayleo's" the Best Cigarette,
Is a tip that you cannot regret;
And in buying, I'll mention
There's a three-pound-a-week pension.

.....

I agree to accept the Committee's decision as final, and enter the Competition on that distinct understanding.

Please sign here .....

Address Envelope—
The Limerick Committee,
c/o Messrs. J. SAMUDA & CO.,
34, Leadenhall Street, London, E.C.

If more than one Solution is sent write same on a plain piece of paper and pin to this entry form (see Rules).

Figure 1: *J. Samuda & Co.* advertisement.

and such historical practices and precedents. These resonances – between manuscript omissions and deletion tasks, between popular poetry contests and the autoregressive generation of final limerick lines – suggest a rich direction for future media archaeology and historical poetics-inflected inquiries into the history of poetry (and texts generally) and the methodologies of contemporary computational literary studies.

2.5 Dataset

The limericks used for the research in this paper originated as a subset of the content from the website *The Omnificent English Dictionary in Limerick Form* (*OEDILF*). Established in 2004, it is an amateur, crowd-sourced project whose goal is to have at least one limerick for every meaning of every word found in the *Oxford English Dictionary*. User-submitted limericks are subject to approval by moderators, and, if approved, are published on the website.[5] Among the benefits of OEDILF is that it comprises a large number of limericks which can be sorted according to different categories of metadata.[6] On the website, the limericks are organized according to different categories: (a) authors, (b) topics. Last but not least, many printed anthologies of limericks highlight particularly misogynistic and/or racist limericks. While OEDILF has its share of ribald poems, it skews in such a way that it provides a large archive of poems from which we can create a good corpus.

We work from a subset of OEDILF originally gathered in Abdibayev et al. (2021b). Therein, two levels of filtering reduced the original 110,610 published limericks to a set of 65,000. The first level was based on simple structural criteria (limericks must have five lines and must use words – rather than symbols, like emojis or formulae). A next level kept only those limericks that verifiably – by machine – satisfied basic structural properties. Specifically, only limericks where all end-of-the-line words could be verified by machine as satisfying the rhyme scheme were kept. For our fifth line completion task, we only picked limericks whose end rhyme words for the first and second lines could be located within our rhyming dictionary. We then used this set for our beam-search algorithm to generate synthetic fifth lines (see subsection 3.4).

For the deleted words minimal pairs, we further filtered limericks to keep only those where at least three end-of-line words are found in Merriam-Webster's *Collegiate Thesaurus*. This produces 34,699 limericks from which we sampled 10,000 limericks (to reduce computational burden). Later, we used this smaller set as a testing ground for the delayed beam-search task.

3. Experiments and Results

In this section, we set up our experimental procedure and present the results. We first flesh out our experimental design that serves as a framework for the BPoMP tests.

5. See: http://www.oedilf.com/db/Lim.php?View=About.
6. See: http://www.oedilf.com/db/Lim.php.

3.1 General Structure of all BPoMP Tasks

3.1.1 Probability of a Sequence

The BPoMP challenges are probabilistic in nature, in that the final 'judgement' of the machine is a comparison of probabilities derived from a pair of inputs.

In the general schema, a language model G will take as input a variable length sequence of words L, and for each word in the sequence output a probability distribution over the possibilities of a word that it 'thinks' should come next in the sequence. The sample space W in our case is some predefined and finite set of words. The probability distribution refers to the collection of probabilities P for all words in the set of words known to the model, which we call its *vocabulary*.

A sequence L can be as short as a sentence or as long as a paragraph. The model produces a probability for each possible next word in a set of words. The word with the highest probability serves as the model's best 'guess' as to what comes next in the sequence, based on what it has witnessed so far. During training we expose the model to gigabytes of human written text, divided into chunks called training examples, and correct the model's predictions of each word in these training examples – in the sense of modifying an underlying algorithm to give more appropriate probabilities based on the truth – based on words that precede the word in question.

3.1.2 Tokens

The machine works at a slightly finer level of granularity – tokens. In some cases tokens correspond to whole words, while in others (such as ours) they might refer to subword segments, such as *astronomical* being tokenized (that is split into tokens) as *astro* and *nomical*. The vocabulary is then defined as all tokens that we have established through the pre-processing of a – usually very large – training text into subword units by means of a count-based compression algorithm (Sennrich et al. 2015).

3.2 Formal Definition of BPoMP

Having established these concepts, we now can explain BPoMP's structure in finer detail. A language model G takes as an input two sequences S and S^*, which correspond to tokenized limericks L and L^*, and processes each independently, in no particular order. L^* is a transformation of L, or more formally $L^* = f(L)$, where f is a function that alters ('minimally corrupts') the original limerick L. The alterations are aimed at singling out particular linguistic phenomenon associated with limericks.

By processing L, we mean outputting $G(S)$ which will provide a probability P for each token in S, the tokenized version of L. We denote all tokens in the vocabulary of the model as V.

Once we compute the probability for each token in both S and S^*, we compute the total probability of sequences S and S^*, via

$$P(S) = \prod_{i=1}^{|S|} P(w_i | w_{i<})$$

where w_i is the word at a position i and $w_{<i}$ are all words before within the limerick.

Or, rewritten for clarity:

$$P(S) = \prod_{i=1}^{|S|} G(w_1, \dots, w_{i-1}).$$

$G(w_1, ..., w_{i-1})$ is the language model's *estimate* of an abstract 'true' probability of encountering token w_i given words that come before it.

Note that some models, such as BERT, violate this formulation by considering the 'score' of each word by looking both at preceding and succeeding words. There are workarounds to their more exotic ways of computing (Lau et al. 2020), what we can instead call the *pseudo-likelihood* of a word (as these formulations do not satisfy the formal properties of a probability distribution). We will not go into the details of this as it does not contribute to the understanding of our experiments. Bigger models (e.g., GPT-2-medium v. GPT-2) generally give estimates closer to the aforementioned 'true' probability.

In practice, we work with $\log P(S) = \sum_{i=1}^{|S|} \log P(w_i)$ instead of $P(S)$ as it simplifies the computation. However, for simplicity of exposition in the examples we will just use probability P. The beauty of this approach lies in its simplicity and universality across models: Unlike explicit classification, it requires no pre-training and no additional computational units on top of the existing model (adding which may introduce more discrepancies in the comparison between models). Moreover, it opens up possibility of studying how much 'poetic knowledge' a model can learn without being explicitly constructed to do so.

We now can delineate the general structure of any BPoMP task. Given a tokenized, human-written limerick S and its tokenized, automatically generated alteration S^*, we ask a model G to compute $\log P(S)$ and $\log P(S^*)$. Comparing the two numbers tells us whether the model finds the original or the alteration to be more likely. When it deems the original more likely it scores a point. Our overall metric is then a simple accuracy measure: Divide the total points the model scored by the total number of test examples used in the experiment. An example of a BPoMP test point is presented in Figure 2 and Figure 3.

3.3 New BPoMPs

We now present two new BPoMPS that further refine our understanding of capabilities of these large models to encode poetic concepts.

3.4 New BPoMP Challenge 1: Random Word Deletion Task

In this task, we alter the original limerick by deleting M words, for $M \in \{1, 2, 3\}$ (each choice of M is a separate BPoMP task). By 'word' we mean any string that is at least two characters long, separated from other words by spaces. Whenever a word is deleted (whether one or several) its surrounding punctuation is preserved. As noted above, this task somewhat resembles the protocols of erasure poetry.

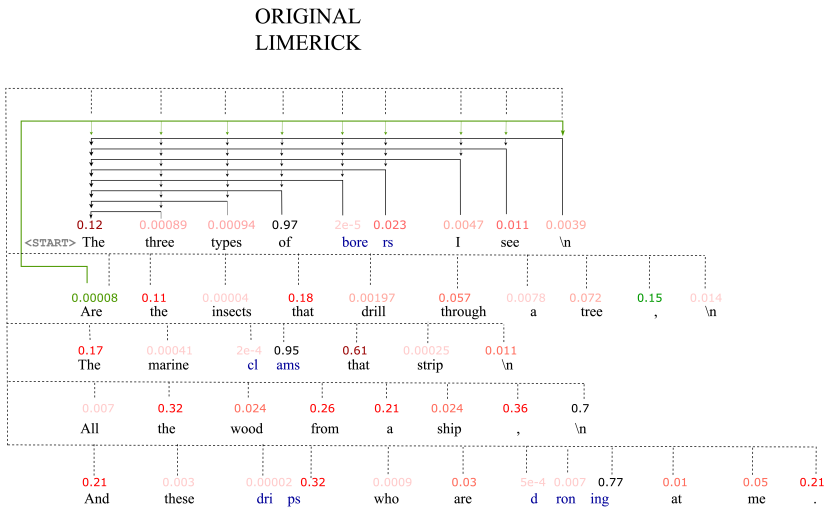


Figure 2: Example of a BPoMP task, specifically, fifth line completion task. Figure 2 is an original, tokenized limerick with each token characterized by its own probability.

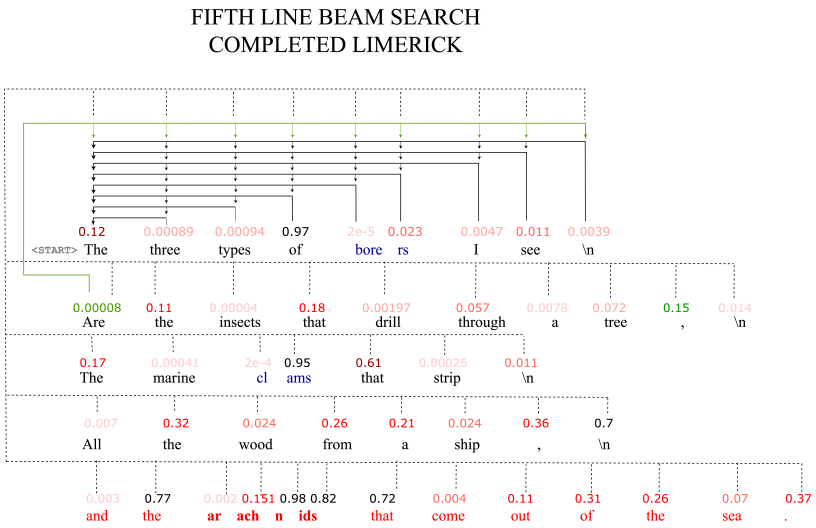


Figure 3: An altered limerick, where the fifth line was replaced by a machine-generated one. All probabilities were produced by GPT-2 medium. The colors of probabilities correspond to magnitude. The arrows represent what words were used by the model when predicting the outputted probability. For simplicity, we didn't include all arrows from the second line onward (with the exception of the first word for explanatory purposes). Every model receives a start token at the beginning of every sequence, for which the probability is not computed (presumed to be 1).

An important consideration is sequence length difference. In this case, the deletion will create a twin that is shorter in word length. This in turn will trivially increase its total probability because the summation of log-probabilities is equivalent to the multiplication of regular probabilities, which in itself is lower for longer sequences on average since we are multiplying numbers between 0 and 1. Thus, to correct for this effect we rescale the total probability of each sequence by length. In other words, we compare

$$P(S_{\text{original limerick}}) = \frac{1}{|S|}\sum_{i=0}^{|S|}\log P(w_i|w_{<i})$$

against

$$P(S_{\text{altered limerick}}) = \frac{1}{|S_{\text{original limerick}}| - M}\sum_{j=0}^{|S_{\text{original limerick}}|-M}\log P(w_j|w_{<j}),$$

This formulation will be used throughout the paper.

	Accuracy		
Model	Delete 1 word	Delete 2 words	Delete 3 words
GPT2	0.89	0.96	0.97
BERT	0.88	0.95	0.97
TransformerXL	0.772	0.8417	0.8816
XLNet	0.4589	0.4412	0.402
Human	**0.95**	**0.975**	**0.925**

Table 1: The BPoMP Word Deletion Test. We delete a varying number (either, 1, 2, or 3) of words (a word is at least two letters long) from a limerick. The task is to pick the original limerick – i.e., to label the text that is most likely to be a limerick. The second line shows the average (two subjects) human (baseline) performance. All tested models easily solve the task.

The results (presented in Table 1) show that most models have no difficulty distinguishing between original limericks and limericks whose semantics were altered by word removals. However, one model remains an outlier: XLNet performs very poorly compared to others. This can be attributed to the somewhat unnatural causal nature of the task.

A second important consideration is the possibility that the three models that perform well on this task (GPT-2, BERT, and TransformerXL) are merely picking up on the grammatical and syntactical conventions they have learned from their training data. In other words, the models are basically functioning as grammar checkers on sentences with missing words. That said, the training data for these models likely include poetic language, and so these models are not only detecting standard prose usage. Still, in order to begin to address this, in the second BPoMP task described below, we test how well models can detect a real limerick ending from a synthetic one – and, in some of those minimal pairs, the limericks exhibit comparable degrees of conformity to what be considered as general and standard English language usage.

	Accuracy					
Model	1-r	1-nr	2-r	2-nr	3-r	3-nr
GPT-2	**0.9133**	0.8991	0.9146	0.9556	0.9163	0.9819
BERT	0.8754	0.8888	0.8727	0.9505	0.8817	0.9801
TransformerXL	0.7464	0.7769	0.7488	0.841	0.751	0.8818
XLNet	0.2897	0.4978	0.2909	0.4931	0.2843	0.4712

Table 2: The BPoMP Rhyming Importance Test. We delete a varying number (either, 1, 2, or 3) of either rhyming or non-rhyming words (a word is at least two letters long) from a limerick. Rhyming word deletion is denoted as $\{1,2,3\}$-r in the Table 2. Non-rhyming deletion is denoted $\{1,2,3\}$-nr. The task is to pick the original limerick – i.e., to label the text that is most likely to be a limerick. By looking at the difference in performance across ten thousand examples, we can theorize on the importance of rhyming words to the models.

Human test subjects were tasked with a slightly different task: to mark if limericks were altered or not. This is due to ease with which one can solve the machine task if presented with both limericks. Their results are uniformly high: Performance dips on three deleted words task, but that can largely be attributed to small sample size, and perhaps, the greater allowance judges made for what passes as a legitimate limerick.

Below is an example of a random deletion task with three words removed from a limerick:

Low to the ground as it goes,
The centipede uses its nose
To find insects to eat,
While an army of feet
Moves what looks like a flexible hose.

Low to the ground as goes,
centipede uses its nose
To find to eat,
While an army of feet
Moves what looks like a flexible hose.

3.4.1 Deleting Rhyming vs Non-Rhyming Words

Here we present an analysis of the effect of removal of rhyming EOL (end-of-line) words as opposed to any other word. We perform the exact same test on all models as in the previous subsection. In the control group, we exclude rhyming EOL words from removal. If rhyming words carry informational importance for models then we expect to see increase in accuracy. Table 2 summarizes the results.

With the exception of GPT-2 in a single scenario, we can conclude that rhyming words *are not* heavily utilized by the models and in fact tend to have lower importance in terms of determining original limericks. This is in line with results obtained in Abdibayev et al. (2021b).

3.5 Transformer Completion Task: Beam Search and Delayed Beam Search

In the Transformer completion task we compute the probabilities comparing an original limerick and its corruption obtained by replacing the fifth line with a line generated by *another* Transformer-based language model (a smaller version of GPT-2 – not one of the models that is being tested) given the first four lines. We ensure that the machine completion necessarily rhymes with the first two lines according to the limerick rhyming scheme. In human evaluation, participants picked out the machine-completed line in all 40 of the test pairs.

3.5.1 Exact Search

To complete the fifth line using a causal model of language (such as GPT-2), we use an algorithm called *search*, but more specifically, a variation on it called *beam search* that we will outline in subsubsection 3.5.2.

To understand this algorithm, we need to explain its purest form first – *exact search*. Exact search works by considering every possible combination of tokens and computing the sequence probability score for each. We then can choose the most likely sequence of a desired length N using our trained models. The caveat is that we cannot exactly compute the most likely sequence because the size of the search space grows exponentially at each step.

As illustration, consider generating a most likely sequence of length 5, such as "Jane visited during the snowstorm". For a vocabulary of size 50,265, the exact search for the most likely sequence has the following total cost: The first step would take 50,265 searches, the second 2.5 billion $(50,256^2)$ searches, the third 126 trillion $(50,256^3)$ searches, the fourth $6.38e^{18}$ $(50,256^4)$ searches and the fifth $3.21e^{23}$ $(50,256^5)$, which is one order of magnitude less than a trillion of trillions of steps.

On average, the fifth line of a limerick in our dataset is ten tokens long (standard deviation 2.14). Thus, we had to consider different, less computationally intensive methods.

3.5.2 Beam Search

Beam search is an approximation technique that restricts the search terms to only a fixed number (k) of sequences at a time, rather than all of them at once.

The first step is the simplest: On input of an initial sequence – the first four lines of the limerick – the model produces a probability distribution (for the next word) over all words in the vocabulary, V. We rank them, with the k highest probability words making it through the first round. In the second step, for each of the k words we compute the probability for every possible next word in the vocabulary. With size of vocabulary $|V|$ (50,256 word fragments for GPT-2) and k words the total cost is $k|V|$. Similarly, in every subsequent step, we generate a probability distribution over the entire possible vocabulary for each of the k surviving words (or more generally, word sequences) and then we re-rank the resulting $k|V|$ sequences to leave only k most likely ones once again. This is beam search. In practice, previous work (Shaham and Levy 2021) has found that beam search performs surprisingly well despite its limited 'field of view'.

Nevertheless, beam search can suffer from a tendency to produce k non-diverse sequences (Holtzman et al. 2019): That is, all of the highly probable outputs are very similar, differing only in one or two words. It rarely ever explores the defined vocabulary, preferring to generate articles ('the', 'a', etc.), as they are very frequently encountered in any text, meaning that it almost never ends up producing a line that ends with a rhyming word for many limericks. We find that we need to use k of size 500-700 to produce rhyming completions. Thus, to get as many limerick fifth line completions as possible we ran this process for a set of 64,872 limericks which only yielded us 5,330 (8%) completions for original limericks. The number is substantially higher (14,155) if

Model	**GPT-2 medium**	BERT	TransformerXL	XLNet (causal)	Human
Accuracy	0.0736	0.1478	0.0253	**0.415**	1

Table 3: Accuracy for the 'Fifth Line Replacement BPoMP Test'. In this test a new fifth line for a limerick is generated by a neural network model (base GPT-2, 142 million parameters) given the original four lines. We generated the completions using the base GPT-2 model after using beam search with results that end with a rhyming word. To preserve the quality of our task, these completions have not been selected for a high probability of the sequence but tested on our models regardless of their absolute probability as determined by base GPT-2 that generated them. The limerick with the highest score as per the calculation is then 'picked' by the machine as the original. The last column shows the average (two subjects) human performance on the task of distinguishing the original limerick from the corrupted limerick. The results demonstrate that models do not perform well at picking out machine completed limericks, while humans have no trouble with the task.

we count all completions of the same limerick that vary by end-of-line rhyme word. We tested models with a sample of 10,000 pairs of original-completion pairs. Separately, we sampled 20 pairs of distinct limericks (that is there were no same limericks in the sample) and presented them to our human test subjects.

We accept the generated line as final if (1) we produce at most N tokens (a number slightly higher than number of tokens in the original fifth line, typically by 2), (2) it ends with a sentence stopping symbol, such as a period or an exclamation mark. The second requirement serves as a partial stoppage in the stream of generation, but does not necessarily mean that the model does not intend to keep going. This can be verified if a model produces its own generation stop symbol (different from grammatical sentence stoppage symbols) that in turn tells us that the model does not intend to continue a sequence.

As Table 3 shows, models have a very strong preference for machine-generated completions, while humans do not. If completions were perfectly comparable to human written ones, we'd expect parity between human and machine performance (floating around 0.5 mark). However, XLNet remains an outlier showing somewhat strong performance. The fact that humans perform well on this task while machines struggle suggests that humans have higher tolerance for what's considered 'valid' language. This further suggests that the underlying model may need to incorporate a more expansive view of what it means to 'learn language' in order to 'understand' or even identify poetry.

Below are two examples of originals and minimally altered limericks:[7]

Example #1:

In my favourite recipe book
Every dish has a photo. I look
At the words (on the page
The pic faces) to gauge
How to roast, boil or fry what I cook.

In my favourite recipe book
Every dish has a photo. I look
At the words (on the page
The pic faces) to gauge
the amount of time it takes for a dish to cook.

7. The altered limericks only appear to have an extra line, and do not actually have an extra line.

Example #2:

Said a guy whose divorce just went through,
"I'm so lucky to bid you adieu.
Best of all is I won
At the lotto, and hon,
I don't need to share any with you."

Said a guy whose divorce just went through,
"I'm so lucky to bid you adieu.
Best of all is I won
At the lotto, and hon,
I'm so delighted to have you back." "Thank you."

3.5.3 Delayed Beam Search

To remedy the problem of certain limericks never yielding results using beam search and specifically to produce diverse solutions, we utilize *Delayed Beam Search* (DBS) (Massarelli et al. 2019). DBS samples the first few words (a number we choose empirically; in our experiments the number is 3) using top-p sampling (Holtzman et al. 2019) and then switches to a regular beam search. Top-p sampling works by reducing the probabilities of all words whose value falls outside of a cumulative range of probability p (a hyperparameter we set), normalizing the rest to sum to 1, and then simply sampling non-uniformly using these (newly normalized) probabilities. After generating the top k sequences using this algorithm, we check if the last word rhymes with the end rhyme words of the first and second lines. If so, we accept this completion as valid. Note that in this approach several differing completions of the same limerick can make it into our test set. We run this algorithm on average 1,000 times for each limerick and add any valid completions generated during this process, before proceeding to the next limerick. Similarly, in this task we are not guaranteed to produce a completion since it is initially stochastic. When we sample the words we simply pick them from a set using probabilities that the model provides. Due to the costs involved in running this process, we restrict the completion generation to a smaller (compared to beam search) set of limericks. Importantly, in our experiments the beam search part uses $k = 5$.

The motivation behind the DBS approach is two-fold: (1) As mentioned above, regular beam search tends to produce non-diverse text and thus, rarely generates a sequence that contains a valid end rhyme, and (2) top-p sampling tends to produce text that often seems unrelated to its preceding context. That said, combining the two helps to alleviate the drawbacks that we see when either is used alone (Massarelli et al. 2019). We additionally cleaned completions of some garbage symbols generated by the model (such as a newline) and then put these limericks into a test set. After using this approach on 10,000 limericks we generated 4,014 test examples derived from 2,595 original limericks. Below is one example:[8]

The Absolute: what do we feel
From the Absolute? Not a great deal.
Our emotional scenes
Are directed by genes,
and we've worked hard to make them feel like
they're real.

The Absolute: what do we feel
From the Absolute? Not a great deal.
Our emotional scenes
Are directed by genes,
and such things are not theirs to reveal.

8. Note that on the lefthand side, the last line extends to include those last two words – i.e., it is not a six-line poem. The righthand side is the original, correct version.

	Model				
	GPT-2 medium	BERT	TransformerXL	XLNet (causal)	Human
Accuracy	0.17	**0.42**	0.07	0.12	1

Table 4: Accuracy for the "Fifth line replacement BPoMP Test using Delayed Beam Search". In this test a new fifth line for a limerick is generated by a neural network model (base GPT-2, 142M parameters) given the original four lines. The last column shows the average (3 subjects) human performance on the task of distinguishing the original limerick from the corrupted limerick.

Another example:

Uncle Ed had repaired to his bed
With a terrible pain in his head,
And by noon he was dead—
So the coroner said—
'Cause his cerebral artery bled.

Uncle Ed had repaired to his bed
With a terrible pain in his head,
And by noon he was dead—
So the coroner said—
That would be I, if he had not been dead.

The results for this task are presented in Table 4. The language models perform poorly, but not as much as in the previous task, going from 16% average performance (across all models) in the previous task to 19%. In particular, GPT-2-medium correctly identifies 17% of the test cases, as opposed to 7% it did prior.

We believe this is a byproduct of the beam search procedure. By sampling the first three tokens we do not overfit to the model's preferences. Moreover, since DBS partially samples the completions, the resulting lines were notably poorer in terms of proper grammar and did not follow the logic of the previous lines as closely as the lines generated by a regular beam search. We stress that we never compared these to human written fifth lines during generation, so they were never filtered to beat (i.e., be more probable under the model) the original completions. In turn, a possible explanation is that the models' statistical 'preferences' (those decoded by the beam search procedure) differ from linguistic preferences of humans, at least when not explicitly trained on poetry.

We presented our human test subjects with 20 samples from the generated set of completed limericks accompanied by their originals. All human judges scored perfectly on the test. Our explanation for this is straightforward: the completions generated by the model tend to be visibly longer than the typical completions written by a human (average of 11 words with std of 2.73 compared to 7 words with std 1.23). This suggests that the poetic knowledge of language models still have some way to go in terms of sensing coherence, a punchline, poetic closure, and meaning generally.

4. Future Work

4.1 Rhyme Probability and Artistry

The second BPoMP challenge necessitated generating synthetic fifth lines, only a percentage of which had correct end rhymes to match lines 1 and 2 of the source limerick (given the *aabba* rhyme scheme). In future work, we hope to explore more minutely how language models fare in generating different kinds of rhyme words given a certain initial rhyme, and the broader implications of *improbable* or difficult rhyme words for

poetic artistry.

4.2 Poetic Minimal Pairs Examples

The investigation into the poetic knowledge of language models approaches poeticity or literariness using a novel approach. We anticipate future work expanding the BPoMP framework to other kinds of poems beyond the limerick and to other poetic features. Below are a next 'level' of examples of minimal pairs. Put aside, for now is the issue of preparing such sets a necessary sub-step that surfaces its own interesting set of challenges in computational poetics.

Ballad or Common Meter

Ballad or Common Meter (four-line stanza, with two pairs of a line of iambic tetrameter followed by a line of iambic trimeter).

Emily Dickinson original vs. minimally flawed example (syllable count).

Original:

> Great streets of silence led away
> To neighborhoods of pause –
> Here was no notice – no dissent –
> No universe – no laws.

Minimally flawed example:

> Great streets of silence led away
> To neighborhoods of pause –
> Here was no notice – no **resistance** –
> No universe – no laws.

Strong Stress (aka Accentual Meter)

Each line has the same number of stresses regardless of the total number of syllables per line. The example is from Samuel Taylor Coleridge's *Christabel* [1816], where every line in the poem has four accents (with a variable number of total syllables per line):

Original:

> The **night** is **chill**, the **cloud** is **gray**:
> 'Tis a **month** be**fore** the **month** of **May**...

Minimally flawed (has extra stress in the second line):

> The **night** is **chill**, the **cloud** is **gray**:
> 'Tis **ma**ny **months** be**fore** the **month** of **May**...

Iambic Pentameter

Iambic pentameter (from Tennyson, *Ulysses*):

Original:

...Made **weak** by **time** and **fate**, but **strong** in **will**
To **strive**, to **seek**, to **find**, and **not** to **yield**.

Minimally flawed (final foot of line two is a trochee):

...Made weak by time and fate, but strong in will
To strive, to seek, to find, and not perish.

Rhyme (from Thomas Gray, *Elegy Written in a Country Churchyard*)

Original:

Full many a gem of purest ray serene,
The dark unfathom'd caves of ocean bear:
Full many a flow'r is born to blush unseen,
And waste its sweetness on the desert air.

Minimally flawed example (the fourth line's end rhyme has been altered with a non-rhyme):

Full many a gem of purest ray serene,
The dark unfathom'd caves of ocean bear:
Full many a flow'r is born to blush unseen,
And waste its sweetness on the desert **sand**.

Rhyme in a Limerick (Edward Lear, *There Was an Old Man with a Beard*)

Original:

There was an Old Man with a beard,
Who said, "It is just as I feared! –
Two Owls and a Hen, four Larks and a Wren,
Have all built their nests in my beard!"

Minimally flawed example (the third line's internal "Hen"–"Wren" rhyme has been disrupted by "Crow"):

There was an Old Man with a beard,
Who said, "It is just as I feared! –
Two Owls and a Hen, four Larks and a **Crow**,
Have all built their nests in my beard!"

Assonance (from John Keats, *Ode on a Grecian Urn*)

Original (recurring long 'i's):

Thou still unravished br**i**de of qu**i**etness,
Thou foster-ch**i**ld of s**i**lence and slow t**i**me...

Minimally flawed example:

> Thou still unravished bride of quietness,
> Thou foster-child of **muteness** and slow time…

Alliteration (from Shakespeare, *Sonnet #30*)

Original (recurring sibilants):

> When to the sessions of sweet silent thought
> I summon up remembrance of things past

Minimally flawed example (in the second line, "summon" is replaced by "conjure"):

> When to the sessions of sweet silent thought
> I **conjure** up remembrance of things past

Consonance (from W.H. Auden, *That Night when Joy began*)

Original (consonance in "flush" and "flash"):

> That night when joy began
> Our narrowest veins to **flush**,
> We waited for the **flash**
> Of morning's levelled gun.

Minimally flawed example (the "flush" – "flash" consonance is disrupted):

> That night when joy began
> Our narrowest veins to flush,
> We waited for the **blaze**
> Of morning's levelled gun.

Imagery and Meaning (from Elizabeth Bishop, *Pink Dog*)

Original:

> Oh, never have I seen a dog so bare!
> Naked and pink, without a single hair…
> Startled, the passersby draw back and stare.

Minimally flawed example (in the second line, the imagery is made less consistent by replacing "hair" with "care"):

> Oh, never have I seen a dog so bare!
> Naked and pink, without a single **care**…
> Startled, the passersby draw back and stare.

Chiasmus and Meaning (from Emily Dickinson, *Much Madness is divinest Sense*)

Original:

> Much Madness is divinest Sense –
> To a discerning Eye –
> Much Sense – the starkest Madness...

Minimally flawed example (in the third line, the parallelism and meaning are disrupted by replacing "Sense" with "Nonsense"):

> Much Madness is divinest Sense –
> To a discerning Eye –
> Much **Nonsense** – the starkest Madness...

5. Conclusion

In this paper, we reported on our experiments in computational poetics with the limerick, thereby continuing its use as a 'model organism' for the discipline. Namely, we presented the formulation of and outcome of two tests constructed using the 'minimal pairs' experimental method for poetry (BPoMP), which are designed to probe the extent to which language models can classify good limericks from slightly altered ones. The language models performed quite well in the first challenge, where an original limerick was compared with its 'corrupted twin', the same but with a few words omitted (which had the effect of disrupting the poem's grammar, syntax, and meaning). In the second challenge, we gave language models a choice between an original limerick and the same limerick except the latter's fifth line now given by a plausible machine-generated replacement for the original final line. On this task, models demonstrate much room for improvement.

Both BPoMP challenges raise all manner of interesting questions about models and their ability to detect human-generated verse from computer-generated verse; resemblances between these tasks and methods of textual criticism, erasure poetry, and the history of the limerick form; and more. Our experiments also point us to future avenues of inquiry, including additional minimal pair challenges that isolate different features of poetry, rhyming artistry, and other unexpected resonances and challenges at the intersection of language models, textual criticism and literary history and analysis.

6. Data Availability

Data can be found here: https://zenodo.org/record/7299879

7. Author Contributions

Almas Abdibayev: Conceptualization, Data Analysis and Preparation, Programming, Writing – original draft

Yohei Igarashi: Poetics, Literary Criticism, Data Analysis, Writing – original draft

Allen Riddell: Methodology, Conceptualization, Data Analysis and Preparation, Writing – original draft

Daniel Rockmore: Methodology, Conceptualization, Data Analysis, Writing – original draft

References

Abdibayev, Almas, Yohei Igarashi, Allen Riddell, and Daniel Rockmore (2021a). "Automating the Detection of Poetic Features: The Limerick as Model Organism". In: *Proceedings of the 5th Joint SIGHUM Workshop on Computational Linguistics for Cultural Heritage, Social Sciences, Humanities and Literature*. Association for Computational Linguistics, 80–90. 10.18653/v1/2021.latechclfl-1.9.

Abdibayev, Almas, Allen Riddell, and Daniel Rockmore (2021b). "BPoMP: The Benchmark of Poetic Minimal Pairs – Limericks, Rhyme, and Narrative Coherence". In: *Proceedings of the International Conference on Recent Advances in Natural Language Processing (RANLP 2021)*. INCOMA Ltd., 1–9. https://aclanthology.org/2021.ranlp-1.1 (visited on 04/22/2022).

American Poets, Academy of (2022). *Erasure | Academy of American Poets*. https://poets.org/glossary/erasure (visited on 12/13/2021).

Ankeny, Rachel A. and Sabina Leonelli (2020). *Model Organisms*. 10.1017/9781108593014.

Anttila, Arto and Ryan Heuser (2016). "Phonological and Metrical Variation across Genres". In: *Proceedings of the Annual Meetings on Phonology* (3). 10.3765/amp.v3i0.3679.

Bode, Katherine (2018). *A world of fiction: digital collections and the future of literary history*. University of Michigan Press.

Brown, Tom B, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. (2020). "Language models are few-shot learners". In: *arXiv preprint*. 10.48550/arXiv.2005.14165.

Chen, Stanley, Douglas Beeferman, and Ronald Rosenfeld (1998). *Evaluation Metrics For Language Models*. https://www.cs.cmu.edu/~roni/papers/eval-metrics-bntuw-9802.pdf (visited on 04/22/2022).

Clark, Elizabeth, Tal August, Sofia Serrano, Nikita Haduong, Suchin Gururangan, and Noah A. Smith (2021). "All That's 'Human' Is Not Gold: Evaluating Human Evaluation of Generated Text". In: *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Association for Computational Linguistics, 7282–7296. 10.18653/v1/2021.acl-long.565.

Dai, Zihang, Zhilin Yang, Yiming Yang, Jaime Carbonell, Quoc V Le, and Ruslan Salakhutdinov (2019). "Transformer-xl: Attentive language models beyond a fixed-length context". In: *arXiv preprint*. 10.48550/arXiv.1901.02860.

Devlin, Jacob, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova (2018). "Bert: Pre-training of deep bidirectional transformers for language understanding". In: *arXiv preprint*. 10.48550/arXiv.1810.04805.

Fujimura, Joan H (1992). "Crafting Science: Standardized Packages, Boundary Objects and 'Translation'". In: *Science as Practice and Culture*. Ed. by A Pickering. University of Chicago Press, 168–211.

Ghazvininejad, Marjan, Xing Shi, Jay Priyadarshi, and Kevin Knight (2017). "Hafez: an Interactive Poetry Generation System". In: *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics-System Demonstrations*, 43–48.

Goldberg, Yoav (2017). *Neural Network Methods in Natural Language Processing*. Morgan & Claypool Publishers.

Gurney, Kevin (1997). *An Introduction to Neural Networks*. UCL Press.

Holtzman, Ari, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi (2019). "The curious case of neural text degeneration". In: *arXiv preprint*. 10.48550/arXiv.1904.09751.

Houston, Natalie M. (2014). "Toward a Computational Analysis of Victorian Poetics". In: *Victorian Studies* 3 (56), 498–510. 10.2979/victorianstudies.56.3.498.

Jurafsky, Dan and James H. Martin (2021). *Speech and Language Processing*. 3rd ed. draft. Prentice Hall.

Lau, Jey Han, Carlos Armendariz, Shalom Lappin, Matthew Purver, and Chang Shu (2020). "How Furiously Can Colorless Green Ideas Sleep? Sentence Acceptability in Context". In: *Transactions of the Association for Computational Linguistics* 8, 296–310. 10.1162/tacl_a_00315.

Lau, Jey Han, Trevor Cohn, Timothy Baldwin, Julian Brooke, and Adam Hammond (2018). "Deep-speare: A joint neural model of poetic language, meter and rhyme". In: *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, 1948–1958. 10.18653/v1/P18-1181.

Legman, Gershon (1969). *The Limerick : 1700 examples, with notes, variants, and index*. Bell Pub. Co.

Liu, Yinhan, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov (2019). "RoBERTa: A Robustly Optimized BERT Pretraining Approach". In: *arXiv preprint*. 10.48550/arXiv.1907.11692.

Long, Hoyt (2021). *The values in numbers: reading Japanese literature in a global information age*. Columbia University Press.

Long, Hoyt and Richard Jean So (2016). "Literary Pattern Recognition: Modernism between Close Reading and Machine Learning". In: *Critical inquiry* 2 (42), 235–267.

Massarelli, Luca, Fabio Petroni, Aleksandra Piktus, Myle Ott, Tim Rocktäschel, Vassilis Plachouras, Fabrizio Silvestri, and Sebastian Riedel (2019). "How decoding strategies affect the verifiability of generated text". In: *arXiv preprint*. 10.48550/arXiv.1911.03587.

McCulloch, Warren S. and Walter Pitts (1958). "A logical calculus of the ideas immanent in nervous activity". In: *Bulletin of Mathematical Biophysics* 4 (5), 115–133.

McInerney, Vincent (2001). *Writing for radio*. Manchester University Press.

Moretti, Franco, ed. (2017). *Canon/Archive: studies in quantitative formalism from the Stanford Literary Lab*. n+1 Foundation.

Piper, Andrew (2018). *Enumerations: data and literary study*. The University of Chicago Press.

Poovey, Mary (2001). "The model system of contemporary literary criticism". In: *Critical Inquiry* 27 (3), 408–438.

Porter, J.D. (2018). *The Space of Poetic Meter*. https://litlab.stanford.edu/hooddistance/ (visited on 04/22/2022).

Preminger, Alex, Terry V.F. Brogan, and Frank J. Warnke, eds. (1993). *New Princeton Encyclopedia of Poetry and Poetics*. Princeton University Press.

Radford, Alec, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. (2019). "Language models are unsupervised multitask learners". In: https://cdn.openai.com/better-language-models/language_models_are_unsupervised_multitask_learners.pdf (visited on 11/11/2022).

Reynolds, L. D. and N. G. Wilson (1991). *Scribes and scholars: a guide to the transmission of Greek and Latin literature*. 3rd ed. Clarendon Press and Oxford University Press.

Rosenblatt, Frank (1943). "The Perceptron: A Probabilistic Model for Information Storage and Organization in the Brain". In: *Psychological Review* 6 (65), 386–408.

Sennrich, Rico, Barry Haddow, and Alexandra Birch (2015). "Neural machine translation of rare words with subword units". In: *arXiv preprint*. 10.48550/arXiv.1508.07909.

Shaham, Uri and Omer Levy (2021). "What Do You Get When You Cross Beam Search with Nucleus Sampling?" In: *arXiv preprint*. 10.48550/arXiv.2107.09729.

So, Richard Jean (2020). *Redlining culture: a data history of racial inequality and postwar fiction*. Columbia University Press.

Underwood, Ted (2019). *Distant horizons: digital evidence and literary change*. The University of Chicago Press.

Vaswani, Ashish, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin (2017). "Attention is all you need". In: *Advances in neural information processing systems*. 30, 5998–6008.

Warstadt, Alex, Alicia Parrish, Haokun Liu, Anhad Mohananey, Wei Peng, Sheng-Fu Wang, and Samuel R. Bowman (2020). "BLiMP: The Benchmark of Linguistic Minimal Pairs for English". In: *Transactions of the Association for Computational Linguistics* (8), 377–392. 10.1162/tacl_a_00321.

Yang, Zhilin, Zihang Dai, Yiming Yang, Jaime Carbonell, Ruslan Salakhutdinov, and Quoc V Le (2019). "Xlnet: Generalized autoregressive pretraining for language understanding". In: *arXiv preprint*. 10.48550/arXiv.1906.08237.