BPoMP: The Benchmark of Poetic Minimal Pairs – Limericks, Rhyme, and Narrative Coherence

Almas Abdibayev
Department of Computer Science
Dartmouth College
Hanover, NH 03755
almas.abdibayev.gr@dartmouth.edu

Allen Riddell
Department of English
Indiana University
Bloomington, IN 47408
riddella@indiana.edu

Daniel Rockmore
Department of Computer Science
Dartmouth College
Hanover, NH 03755
daniel.n.rockmore@dartmouth.edu

Abstract

We adapt BLiMP (Benchmark of Linguistic Minimal Pairs) language model evaluation framework to the context of poetry, introducing the first of a series of tasks titled Benchmark of Poetic Minimal Pairs (BPoMP). The tasks presented herein use one genre of English-language poetry, the limerick (five-lines, rhyme scheme AABBA). Following the BLiMP schema, the BPoMP tasks use 10,000 minimal pairs of limerick/corrupted limerick. The latter is created by (1) shuffling two rhyming end-of-the-line words, (2) shuffling two rhyming lines, (3) replacing end-of-the-line word by a non-rhyming synonym. Our general task is detection of the original limerick, which we believe tests a language model's capacity to utilize "end rhymes", a common feature of poetry. We evaluate Transformer-based models by checking if they assign a higher probability to the non-corrupted limerick in each minimal pair. We find that the models identify the original limerick at rates better than chance, but with a nontrivial gap relative to human accuracy (average of 98.3% across tasks). The publicly available curated set of limericks accompanying this paper is an additional contribution. In general, we see this as a first step to create a community of NLP activity around the rigorous computational study of poetry.

1 Introduction

Machines — i.e., artificial intelligence – continue to make great progress toward the challenges of machine-assisted and machine-written literature. Examples range from word auto-completion algorithms to deep learning-based methods that (sometimes) produce Turing Test-passing text (Elkins and Chun, 2020) and even credible sonnets (Ghazvininejad et al., 2016). Evaluating such work uses a range of techniques, including automatic metrics such as BLEU as well as – rightly or wrongly – Turing Test-inspired human evaluation.

This paper focuses on testing if large models pretrained on gigabytes of text, such as GPT-2 (Radford et al., 2019) and Transformer-XL (Dai et al., 2019)), are capable of "discovering" the rhyming and basic narrative information inherently present within a simple poetic form, the limerick, a five-line poem, usually humorous, with rhyme scheme AABBA and regular – albeit not fixed – metrical structure. In particular, we test Transformer models on various (suitably contextualized) forms of the line completion problem for the limerick as well as a fundamental narrative-oriented task. These tests should be viewed as a first step toward a broader goal of framing the rigorous interrogation of the poetic (or more broadly, literary) capabilities of language models. Given their formulaic structure (e.g., AABBA rhyming scheme) and the relatively small size of each unit, limericks present a good testing ground as compared to longer or more complex poems, such as Shakespearean sonnets (see e.g., (Ghazvininejad et al., 2016)), or most broadly, "literature".

To this end we gathered a small data set of 99,000 limericks (filtered down further for quality purposes), which we use to test three poetic feature: understanding the concept of a rhyme framed as two kinds of line completion test, and one test of narrative structure, framed as a line ordering test, given the full limerick. The primary contri-

Proceedings of Recent Advances in Natural Language Processing, pages 1–9
Sep 1–3, 2021.
https://doi.org/10.26615/978-954-452-072-4_001

butions of this paper are the instantiations of these tests as "minimal pair" tasks and a curated and publicly available dataset of limericks. The former are designed in the spirit of the BLiMP (Benchmark of Linguistic Minimal Pairs) tests (Warstadt et al., 2020) that are used to interrogate more general linguistic capabilities of language models. Our limerick-based test is a first construction that will be part of a larger collection of tests of machine poetry capabilities, given the title BPoMP, or Benchmark of Poetic Minimal Pairs. As for the latter contribution we hope that the availability of this curated data set inspires other computational studies of this and other poetic forms.

Recognizing the basic ingredients of a limerick (or any poetic form) are also part of being able to produce a limerick and thus, we also see this as a first step toward another larger goal: machine composition of high-quality limericks, or any prescribed poetic form.

2 Related Work

This paper is a part of the growing body of literature devoted to the formulation and execution of tasks for probing language models, including the "BERTology" literature that focuses on BERT (see e.g., (Tenney et al., 2019; Michel et al., 2019; Clark et al., 2019; Hewitt and Manning, 2019). In formulation, it is directly modeled on the BLiMP framework (Warstadt et al., 2020), wherein the machine is a given a pair of instances, one correct, and another "minimally" modified (i.e., "corrupted" in some very minor, but systematic way) and the language model then outputs probabilities (or some derivative thereof) signifying which is the most likely. Our line completion tasks produce line completion-based metrics, and thus are related to the large body of work already devoted to the sentence completion task (see e.g., (Mirowski and Vlachos, 2015)). Our simple exploration of narrative structure, interrogated through the minimal pair of limericks, one an original and the other with two rhyming lines swapped, is related to, but different from work on next sentence prediction (Cui et al., 2018), which presents two sentences of text in original and swapped orders (but without the fuller context that our minimal pair of limericks provides) and produces probabilities for both occurrences.

Our work also drew partial inspiration from poetry generation literature (Ghazvininejad et al., 2016; Li et al., 2018; Liu et al., 2019; Lau et al., 2018). However, in this work we do not aim to create machine poetry, but rather show the way in which poetry can be used as a prism to evaluate models which purport to be accurate models of a wide variety of human-produced texts, i.e. texts written in a variety of genres and registers.

3 Problem Statement

In the spirit of BLiMP (Warstadt et al., 2020), we present the language model with a five-line limerick and its corrupted counterpart and compare the probabilities the model assigns to each of these text blocks. Note that for each test we always include a full limerick and full limerick with corruption (which thus may or may not still be a limerick). The corrupted limerick differs from its source in several possible ways (in the order of increasing difficulty): (1) two rhyming words swap places, (2) two rhyming lines swap places, (3) one word is changed, always an end-of-the-line rhyming word which has been replaced by a synonym (see below for details) that removes rhyming information. Each of the three defines its own task. The difficulty ordering is based on the amount of corruption introduced into the limerick—from somewhat obvious to more subtle. Note however, that this doesn't necessarily correspond to empirical human rating of difficulty. We mostly focus on causal models, which allows us to infer the probability of sequence (i.e. limerick), by multiplying estimated probabilities of each token using previously seen tokens as context (in practice we sum log-probabilities) (Bengio et al., 2003):

$$P(S) = \prod_{i=0}^{|S|} P(w_i|w_{<i})$$

where S is the limerick sequence and w_i is the token at the position i.

In some cases (i.e. BERT) we use a modified pseudo-likelihood formulation we adapted from (Lau et al., 2020), to compute "probabilities" using bidirectional context:

$$P(S) = \prod_{i=0}^{|S|} P(w_i|w_{<i}, w_{>i})$$

This produces a simple experiment (and 3 tasks): given two sequences, a limerick and its corrupted

Original	Shuffled rhyming word	Shuffled line	Corrupted rhyming word
Television or radio stations	Television or radio **nations**	Television or radio stations	Television or radio stations
May broadcast to several nations.	May broadcast to several **stations**.	May broadcast to several nations.	May broadcast to several **countries**.
One can tell them apart	One can tell them apart	Through the call letters' art—	One can tell them apart
Through the call letters' art—	Through the call letters' art—	One can tell them apart	Through the call letters' art—
Which provides their identifications.	Which provides their identifications.	Which provides their identifications.	Which provides their identifications.

Table 1: Example of a limerick and its corruptions. Each corruption type forms its own task type. Thus, in total we have 3 tasks, but the experiment is the same in each task.

version, the model must correctly "guess" the original by assigning a higher probability to it. Each limerick pair then serves as a single test point. However, the source limerick may appear more than once within the dataset, since we may have generated several corrupted versions.

4 Experimental Setup

4.1 Dataset

We did not source published books of limericks, as they lacked machine-readable editions and tended to use dated language. Instead, we worked from the website *The Omnificent English Dictionary In Limerick Form* (OEDILF). OEDILF[1], established in 2004, publishes user-submitted limericks subject to approval by moderators. The topics are generated by users and moderators. Our scrape of OEDILF contains limericks by 1624 different authors. The distribution of limericks by author is not uniform, with the top 10 authors responsible for ca. 40 % of all limericks. We first filtered limericks based on simple criteria: limericks must have 5 lines and must use words (as opposed to symbols, such as emojis or formulae). This excludes a range of unorthodox limericks such as ASCII art limericks. The result was a corpus of 98,454 limericks (dated 1998–present). For our experiment we further require all "limericks" satisfy some simple machine-identifiable limerick criteria: (a) end-of-the-line words must be in our rhyming dictionary, (b) 2nd and 3rd lines must have fewer syllables than other lines. Our first criterion is there to ensure all our limericks can be identified as following the AABBA rhyme scheme. Our second follows from the conventional definition of a limerick (and is tied to metrical structure). Verification uses CMUdict[2] and a custom character-level RNN model used to predict the number of syllables in the words, trained on CMUdict and syllable count information from Wiktionary[3]. The details of this model can be found in the Appendix. This second stage filtering leaves us with 29,853 limericks to build the set of minimal pairs. This clean subset contains 52,316 unique words, and a total of 879,653 tokens. Next, we discuss the corruption process of the limericks for all of our tasks. Note that in every case, we sample a limerick from the clean subset and pair it up with its corrupted version using procedures described below.

4.2 "Shuffled" Test Sets

Two of our tasks shuffle the contents of the original limerick, by either swapping two words, or swapping two lines. We generated these as follows.

Our first shuffled dataset, samples a limerick and randomly swaps end-of-the-line two rhyming words (as defined by AABBA scheme) within it. Doing so, we aim to preserve rhyming information but also to distort the semantics. Swapped words are checked to belong to the same part of speech during the replacement process. In certain instances, rhyming words can in fact repeat within the same limerick, but we found that these account for only 0.05% of all altered limericks.

Our second shuffled dataset randomly swaps two lines in a sampled original limerick. In this case, we are trying to remove longer-horizon semantic information, as well as to disturb the meter. As previously, they must belong to the same part of rhyming scheme, A or B. We make sure to remove any obvious datapoints, such as lines that start a quotation, or corrupted limericks that now end with a non-line ending symbol, such as comma, as a result of the swap.

For both of these test sets, we sample 10,000 test

[1] http://www.oedilf.com/db/Lim.php
[2] http://www.speech.cs.cmu.edu/cgi-bin/cmudict
[3] https://en.wiktionary.org/wiki/Wiktionary:Main_Page

pairs of original and altered limerick.

4.3 Corrupted Rhyme Test Set

The corrupted rhyming word limerick dataset was auto-generated using the following procedure. For each limerick we replace the ith end-of-the-line rhyming word, $i \in \{1, 2, 3, 4, 5\}$ with a non-rhyming synonym. To generate high-quality synonyms we used Merriam-Webster's Collegiate Thesaurus API.[4] We were able to retrieve synonyms for 34,699 unique end-of-the-line words. Synonyms were then filtered by part of speech (Merriam-Webster provides different synonyms for each POS the query word potentially belongs to and it has in its database[5]). We utilize spaCy (Honnibal et al., 2020) as our POS tagger. Once the set of synonyms is found, we choose the synonym with the closest contextualized representation (within the limerick) to the representation of an original word provided by $\texttt{BERT}_{\texttt{LARGE}}$ (Devlin et al., 2018). Separately, we score each synonym using a different scheme: we replace each end-of-the-line word with a synonym using the same process described above, and then use GPT-2 to score total limerick probability of each potential corrupted limerick. We then choose the highest probability replacement and see if it agrees with the BERT replacement. In case of agreement, we add the corrupted limerick to the pool. We want to stress that at no point are we comparing the corrupted limerick to the original in terms of total sequence probability as this would interfere with our experimental goals. In other words, all comparisons are made between candidate synonym replacements. This GPT-2 filtering ensures that most of our replacements are of high quality. Finally, we exclude all limericks whose tokens are not in any of the models' vocabularies, and then sample 10,000 limerick pairs for the final test set.

4.4 Models

We assess a family of Transformer-based (Vaswani et al., 2017) language models. Transformer is a neural network that consists of several layers of attention (Bahdanau et al., 2014) interspersed with fully connected layers. For all Transformer models we utilize Hugging Face's `transformers` (Wolf et al., 2020) library.

GPT-2 (Radford et al., 2019) is a causal language model, meaning that it predicts a word based only on preceding words. Thus, for line 3, the model uses all words from lines 1 and 2, and potentially uses rhyming information contained within the last word of line 1. We use `GPT-2-Medium` that has equivalent number of parameters (345 million) as all other Transformer models we test. GPT-2 was pretrained on a custom, 40 GB size dataset called WebText, which includes various HTML pages from around the web filtered so as not to include Wikipedia text.

BERT (Devlin et al., 2018) uses bidirectional self-attention, which in addition to looking behind, also looks ahead in sequence. BERT is trained on a masked language task, where in a given sequence (a sentence, a paragraph) certain words are masked at random are then predicted given the surrounding context. We use uncased BERT in the generation of corrupted synonyms, and cased model in our experiments. In both cases we use 340M model. BERT is trained on a 3,300 million word combined corpus of regular prose (BookCorpus, (Zhu et al., 2015)) and an English Wikipedia dump scraped by the authors.

Transformer-XL (Dai et al., 2019) is another causal language model that differs through the addition of a recurrence mechanism: hidden states from previous segments (a fixed-length context window that the model uses to predict the next word) are carried over to the next segment, which theoretically extends the context length that Transformer-XL can utilize. In addition to support this recurrence mechanism the authors propose a novel position embedding scheme. Transformer-XL was pretrained on Wikitext-103 (Merity et al., 2016), a dataset of 103M tokens scraped from English Wikipedia. We use the 340 million parameter version of Transformer-XL.

XLNet (Yang et al., 2019) differs from other models in the optimization objective: the model is trained using a novel permutation language model objective, where instead of a fixed-order (left to right) context, the model is exposed to a randomly permuted sequence, while predicting the last word

[4] `https://dictionaryapi.com/products/api-collegiate-thesaurus`. We publish a static version of the set of minimal pairs so that all results reported here can be verified easily.

[5] For example *eye* returns 'noun': ['ring', 'navel', 'scrutiny', ... , 'consciousness', 'vision', 'loop'], 'verb': ['deliberate', 'sight', 'ponder', 'meditate', 'perceive', ... , 'watch', 'wrestle (with)']

(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. At the same time it is also causal as the generation is still left to right. In addition, XL-Net reuses segment level recurrence as introduced in (Dai et al., 2019) to increase potential context length. XLNet, is trained on BookCorpus, English Wikipedia, but adds heavily filtered versions of datasets Giga5, ClueWeb 2012-B, and Common Crawl[6] (Parker et al., 2011; Callan et al., 2009), resulting in 33 billion subword-piece dataset. The model we test has 340 million parameters.

5 Results

Our general experimental framework is as follows: we ask if the models assign higher probability to the uncorrupted limerick of the pair. Again, in each test the "minimal pair" is a pair of five-line texts, one the original limerick, the other a corruption, which may or may not still be a limerick. We compare our models using simple count-based accuracy: total number of correctly guessed pairs (i.e., pairs where the higher probability is assigned to the uncorrupted limerick) versus total number of (minimal) pairs.

Table 2 summarizes our results. For the task of differentiating between the original limerick and its twin with shuffled rhyming words, GPT-2 wins by a small margin over other Transformer models. The differences are very small, with the exception of Transformer-XL, which performed poorly (a pattern that appears in other tasks). Overall, the models were easily able to distinguish semantically inconsistent corrupted limericks.

Next we take a look at the task of swapping lines that rhyme. Since the semantic issues here are not as apparent, we expected performance of all models to dip compared to our first task. This largely has proven to be true, as accuracy drops by an average of 11% across all models. Here, as well, GPT-2 emerges to be superior in terms of performance.

Finally, we assess results on the most difficult task, where we preserve the semantic information but diminish rhyming information, by replacing one word with a non-rhyming synonym. The accuracy drops significantly, compared to first two tasks, and the gap between human and model performance is considerable. Here, BERT is the clear "winner" is that it outperforms the favorite GPT-2 by almost 5 percentage points. We theorize that the better performance comes from BERT's increased bidirectional context. Based on these three experiments, rhyming information seems to matter to a non-trivial extent for these models.

5.1 Human Baseline

All models perform worse than human judges. Human readers correctly identify the original limerick with probability 1 in shuffled rhyming words and corrupted rhyming word tasks. The subjects emphasized that they identified corrupted limericks after scanning end-of-the-line words to find the non-rhyming word. On the shuffled line task they get 95% accuracy. The human experiments were performed with 11 test subjects in corrupted rhyming word task and 2 test subjects in the remaining two tasks. All test subjects were native English speakers and none poetry experts. Each test subject was presented with 20 randomly sampled pairs of limericks presented in randomized order. Arguably, in the shuffled line test, for both test subjects the mislabeled limerick was still syntactically - and metrically and rhyme scheme correct. We suspect that a similar test performed with a larger sample of human annotators would produce identical results.

5.2 GPT-2 Error Analysis: Synonym Corruption Case Study

For a more detailed error analysis we focus on the GPT-2 results within the corrupted rhyme task as the model is the clearest in terms of optimization task (next-word prediction) and performs well in most tasks. To analyze the GPT-2 errors we considered the distribution of log-probability differences between original and corrupted limericks, identifying tail cases with largest difference.[7] This is a proxy for a measure of model prediction "confidence."

We hypothesize that difference in the n-grams counts[8] ($n \in \{2, 3, 4\}$) used in the last tokens of the corrupted line is strongly correlated with the log probability (the one model assigns to the entire limerick) difference between the limerick pair.

First we consider examples where the model assigned relatively high probability to the correct (original) limerick. We expect to see that the original bi-gram[9] occurs with much higher frequency

[6] https://commoncrawl.org/

[7] See Appendix.

[8] As observed in the Google Ngram dataset: https://books.google.com/ngrams

[9] We start with bi-grams, and in some cases used 3- and

Model	Accuracy		
	Shuffled rhyming words	Shuffled rhyming lines	Corrupted rhyming word
BERT (cased)	0.9414	0.8321	**0.7034**
GPT-2	**0.9423**	**0.8336**	0.6557
Transformer-XL	0.8724	0.6623	0.6188
XLNet	0.9310	0.8227	0.6741
Human	**1.0000**	**0.9500**	**1.0000**

Table 2: Accuracy in all 3 tasks in which we stress test Transformer models, with last line showing the average (2 subjects) human performance. In shuffled rhyming word task, corrupted limerick has two rhyming end-of-the-line words switch places. Shuffled rhyming lines, as the name implies, switches the position of two lines, e.g. line 1 and 5 trade places and corrupted limerick is rearranged in the line form 5,2,3,4,1. Lastly, in corrupted rhyming word task we choose one end-of-the-line word at random and replace it with a non-rhyming synonym found in Merriam-Webster dictionary.

than the corrupted bi-gram, which would translate to higher difference in bi-gram counts. The Pearson correlation for seven tail end cases was 0.71. This suggests that bi-gram frequency difference is a moderately good predictor of the log-probability difference[10]. Next, we considered examples in which the model was very confident and wrong. In all such cases the GPT-2 output could be explained by bi-gram counts, with correlation of 0.89 for five extreme tail cases. In other words, the corrupted bi-grams displayed much higher frequency than their correct (original) counterparts. Note however, that this proxy falls apart as correlation reduces to -0.39 once we include more cases where the model isn't as confident.[11] Thus, for high confidence scenarios bi-gram statistics explain the differences in probabilities the model assigns to either original or a fake.

6 Conclusion

In this paper we have introduced a first instance of a Benchmark of Poetic Minimal Pairs (BPoMP), a framework for the testing of the poetic "knowledge" of a language model. Using a curated set of limericks we interrogate the abilities of four Transformer-based language models on three poetry-based tasks (end-of-the-line rhyme replacement, rhyming word swapping, and rhymed line swapping) framed as distinguishing a limerick from a minimally corrupted version. In each case the Transformer-based models succeed at a rate better than chance, but much worse than the perfect and near-perfect human baseline. The limerick dataset is made publicly available.

Future near-term contributions to the BPoMP framework include meter, assonance, and alliteration tested in more complex short poetry. While the limerick may appear to be a "simple" textual form, it is in that way a perfect kind of "model organism" for a more general study of poetry and even literature. A five-line poem of this form can still articulate a range of narrative structure and sophisticated literary devices, but is of a size that admits relatively easy modification and interrogation – e.g., through the minimal pairs framework. We hope that this paper and the concomitant data set release inspire such future work.

4-grams.

[10]We find a few outliers (with negative frequency difference) farther from the tail end, where our hypothesis fails. Table 4 – see the Appendix – presents these outliers as well as a 3- and 4-gram comparison that supports our hypothesis.

[11]For correct cases the correlation quickly drops to 0 if log-probability differences close to 12 are included.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin. 2003. A neural probabilistic language model. *The journal of machine learning research*, 3:1137–1155.

Jamie Callan, Mark Hoy, Changkuk Yoo, and Le Zhao. 2009. Clueweb09 data set.

Kevin Clark, Urvashi Khandelwal, Omer Levy, and Christopher D. Manning. 2019. What does BERT look at? an analysis of BERT's attention. In *Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 276–286, Florence, Italy. Association for Computational Linguistics.

Baiyun Cui, Yingming Li, Ming Chen, and Zhongfei Zhang. 2018. Deep attentive sentence ordering network. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4340–4349, Brussels, Belgium. Association for Computational Linguistics.

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

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

Katherine Elkins and Jon Chun. 2020. Can gpt-3 pass a writer's turing test? *Journal of Cultural Analytics*.

Marjan Ghazvininejad, Xing Shi, Yejin Choi, and Kevin Knight. 2016. Generating topical poetry. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1183–1191, Austin, Texas. Association for Computational Linguistics.

John Hewitt and Christopher D Manning. 2019. A structural probe for finding syntax in word representations. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4129–4138.

Matthew Honnibal, Ines Montani, Sofie Van Landeghem, and Adriane Boyd. 2020. spaCy: Industrial-strength Natural Language Processing in Python.

Jey Han Lau, Carlos Armendariz, Shalom Lappin, Matthew Purver, and Chang Shu. 2020. How furiously can colorless green ideas sleep? sentence acceptability in context. *Transactions of the Association for Computational Linguistics*, 8:296–310.

Jey Han Lau, Trevor Cohn, Timothy Baldwin, Julian Brooke, and Adam Hammond. 2018. Deep-speare: A joint neural model of poetic language, meter and rhyme. *arXiv preprint arXiv:1807.03491*.

Juntao Li, Yan Song, Haisong Zhang, Dongmin Chen, Shuming Shi, Dongyan Zhao, and Rui Yan. 2018. Generating classical chinese poems via conditional variational autoencoder and adversarial training. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3890–3900.

Zhiqiang Liu, Zuohui Fu, Jie Cao, Gerard de Melo, Yik-Cheung Tam, Cheng Niu, and Jie Zhou. 2019. Rhetorically controlled encoder-decoder for modern chinese poetry generation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1992–2001.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. 2016. Pointer sentinel mixture models. *arXiv preprint arXiv:1609.07843*.

Paul Michel, Omer Levy, and Graham Neubig. 2019. Are sixteen heads really better than one? In *Advances in Neural Information Processing Systems*, volume 32, pages 14014–14024. Curran Associates, Inc.

Piotr Mirowski and Andreas Vlachos. 2015. Dependency recurrent neural language models for sentence completion. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 511–517, Beijing, China. Association for Computational Linguistics.

Robert Parker, David Graff, Junbo Kong, Ke Chen, and Kazuaki Maeda. 2011. English gigaword fifth edition, linguistic data consortium. Technical report.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. *OpenAI Blog*, 1(8):9.

Ian Tenney, Dipanjan Das, and Ellie Pavlick. 2019. BERT rediscovers the classical NLP pipeline. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4593–4601, Florence, Italy. Association for Computational Linguistics.

Ashish Vaswani, 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*, pages 5998–6008.

Alex Warstadt, 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. *Transactions of the Association for Computational Linguistics*, 8:377–392.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

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

Yukun Zhu, Ryan Kiros, Rich Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In *Proceedings of the IEEE international conference on computer vision*, pages 19–27.

A Appendix

A.1 Compute Resources

For most of the model testing and synonym generation work we used 2 NVidia Titan X graphics cards with 12GB of VRAM, running CUDA 11.2. In parallel, we also used Google Colab with T4 instances that sport 16GB of VRAM and also run CUDA 11.2.

A.2 OEDILF Description

The limericks are organized according to two categories: (a) included words contained and (b) topics. The latter are tags provided by both the managers of the website and the people who submitted the limericks. The dictionary part of the website refers to these words that could be found within a limerick, that is for any word in this dictionary there is a corresponding limerick that contains this word.

A.3 Identifying Number Of Syllables

Not all words in the vocabulary of our dataset are present in CMUDict, which only includes 133,779 examples. The full 98,454 limerick dataset (recall that there were two filtering steps to get us from this full set to the set used in the experiments) produces a vocabulary of 110,650 words, of which 55% are not present in CMUdict. To estimate the number of syllables in the vocabulary we trained a small ($\sim$ 9800 parameters) character-level RNN[12]. The model consists of an embedding layer, a one-layer LSTM, with hidden size of 30, and a final linear layer that outputs back to characters. We train our charLSTM to be a classifier over 20 classes where each class determines the number of syllables in a word. The number 20 was set as an upper bound of number syllables based on the fact that the longest syllable word known in dictionaries has 19 syllables. A simple regression model did not work, nor did it optimize very well, displaying mode collapse-like behavior (i.e., consistent prediction of the average value, which is not useful for our discrete scenario).

[12]We based the code from https://github.com/spro/char-rnn.pytorch

Original	Corrupted
To the town, folks from near and from far Come to shop at the weekly bazaar. Corn and cabbage and milk Can be found beside silk, As much as can fit in the car.	To the town, folks from near and from way Come to shop at the weekly bazaar. Corn and cabbage and milk Can be found beside silk, As much as can fit in the car.

Table 3: Example of bigrams we captured for the analysis. Bigrams are highlighted in color. The limerick comes from a tail case when the model is correctly very confident.

Initially, we trained using pure CMUdict and around 15,000 words were used as a validation set, with another 118,000 used as as a training set. For our test set we decided to predict the number of syllables within lines of 21 limericks (which we labelled manually), as it emulates our scenario more closely than pure prediction of syllables in unseen words. We removed all of the words present in limericks from the training set. After analysis we determined that CMUdict did not contain many words with more than 11 syllables, so we supplemented it with another source: Wiktionary, an open source dictionary with user-submitted definitions of words. Wiktionary contains statistics with regards to number of syllables per word[13] that can be scraped quite easily. We were able to obtain training examples with up to 19 syllables. In total that gave us additional 40789 training examples. We trained the model using ADAM optimizer, setting learning rate to 0.001 until validation loss went below 0.2 (around 10-12 epochs depending on the optimization curve).

A.4 GPT-2 Analysis And Bi-gram Outliers

In this section we present various supporting materials we used in our analysis of cases where the model assigned high probability to a limerick within a test pair. Table 3 presents an example limerick pair from our test set, in which GPT-2 correctly guessed the original with high confidence. We highlight the bigrams we used in our analysis section 5.2.

[13]https://en.wiktionary.org/wiki/Category:English_words_by_number_of_syllables

Ngram	Original	Corrupted
2-gram	your senses 6e-07	your feelings **1.73e-06**
3-gram	to your senses **1.5-07**	to your feelings 6.2e-08
4-gram	come to your senses **9.5e-08**	come to your feelings **NA**
2-gram	your court 1.3e-07	your yard **1.4e-07**
4-gram	ball's in your court **6e-9**	ball's in your yard **NA**
2-gram	a time 4.8e-05	a moment **7.3e-05**
3-gram	at a time **1.8e-05**	at a moment 1e-06
2-gram	the hip 1.8-06	the cool **4.2e-06**
3-gram	from the hip **1.3e-07**	from the cool 7.4e-08
2-gram	for broke 6.9e-08	for poor **1.3e-06**
3-gram	go for broke **4.1e-08**	go for poor 3.7e-10
2-gram	to cheese 4.1e-08	to trash **7.9e-08**
3-gram	due to cheese **2.4e-10**	due to trash 1.2e-10

Table 4: Outliers: target end-of-the-line word and its preceding neighbor taken from original and corrupted limericks which GPT-2 labelled correctly, but whose bi-gram frequency is higher for the corrupted limerick. This violates our hypothesis that bi-gram frequency would be higher for the original limerick. Upon closer examination we realize that 3- and 4-grams still fit our hypothesis. If frequency is listed as **NA** then it was not found in enough documents within Google Ngram corpora. Bigger n-grams (compared across rows) are bolded. The outliers were found when analyzing set of log-probabilities farther from the tail of empirical distribution.

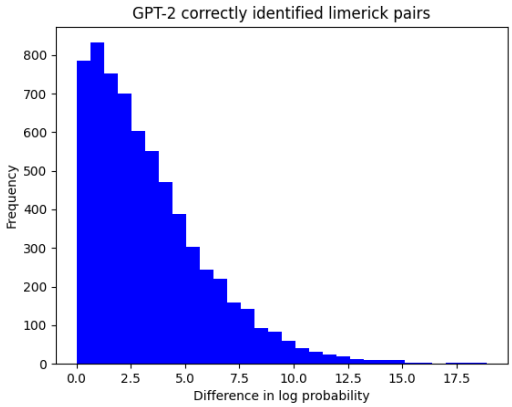

Figure 1: Histogram plotting log-probability difference between original and corrupted limerick. Since we are plotting correctly identified limerick pairs we are taking the difference between $\log P(original) - \log P(corrupted)$ which is positive. Note the tail cases that start around 15.12 mark.

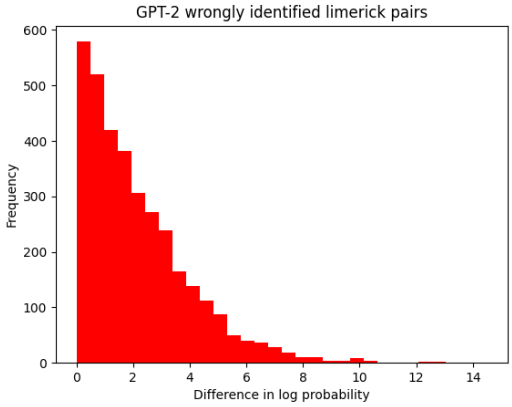

Figure 2: Histogram plotting log-probability difference between original and corrupted limerick. Since we are plotting incorrectly identified limerick pairs we are taking the difference between $\log P(corrupted) - \log P(original)$ which is positive. Note the barely visible tail cases that start around 12 mark.