Train and Constrain: Phonologically Informed Tongue Twister Generation from Topics and Paraphrases

Tyler Loakman[1*], Chen Tang[2*], and Chenghua Lin[2**]

[1]The University of Sheffield, Department of Computer Science
`tcloakman1@sheffield.ac.uk`

[2]University of Manchester, Department of Computer Science
`chen.tang@manchester.ac.uk`, `chenghua.lin@manchester.ac.uk`

*Previous work in phonologically and phonetically grounded language generation has mainly focused on domains such as puns and poetry. In this article, we present new work on the generation of English tongue twisters—a form of language that is required to be conditioned on a phoneme level to maximize sound overlap, while maintaining semantic consistency with an input topic or phrase and still being grammatically correct. We present **TwisterLister**, a pipeline for generating phonologically informed tongue twisters from large language models (LLMs) that we use to generate **TwistList 2.0**, the largest annotated dataset of tongue twisters to date, consisting of 17k+ examples from a combination of human and LLM authors. Our generation pipeline involves the use of a phonologically constrained vocabulary alongside LLM prompting to generate novel, non-derivative tongue twister examples. We additionally present the results of automatic and human evaluation of smaller models trained on our generated dataset to demonstrate the extent to which phonologically motivated language types can be generated without explicit injection of phonological knowledge. Additionally, we introduce a phoneme-aware constrained decoding module (**PACD**) that can be integrated into an autoregressive language model and demonstrate that this method generates good quality tongue twisters both with and without fine-tuning the underlying language model. We also design and implement a range of automatic metrics for the task of tongue twister generation that is phonologically*

* Co-first authors.
**The corresponding author.

Action Editor: Tal Linzen. Submission received: 15 March 2024; revised version received: 15 July 2024; accepted for publication: 20 September 2024.

https://doi.org/10.1162/coli_a_00544

*motivated and captures the unique essence of tongue twisters, primarily based on phonemic edit distance (**PED**).*[1]

1. Introduction

Although the dawn of large language models (LLMs) such as OpenAI's GPT-4 (OpenAI et al. 2023), Meta's Llama 2 (Touvron et al. 2023), and Google's Gemini (Anil et al. 2023) has brought unprecedented performance improvements in many natural language generation (NLG) tasks, these models are highly resource-hungry in terms of data, computation, and API expenses. Consequently, many works have started to investigate the ways in which LLM capabilities and knowledge can be infused into smaller models, using the larger model for data enhancement via the generation of pseudo-labels (Tang et al. 2023) or additional training examples (Yang, Tang, and Lin 2024).

Additionally, LLMs are primarily designed to select the most probable continuation of a span of text based on their training data. Creative language, in direct opposition to non-creative language, is predominantly desired to be non-derivative, containing phrases and word sequences that are not ubiquitous in everyday language in order to evoke various emotions and engage a reader rather than purely convey information in a linguistic form. As a result, creative language generation's goals are at odds with the primary training paradigm of LLMs, as the goal is often *not* to select the most probable continuation, and instead surprise and engage readers (Roush et al. 2022).

In particular, tongue twisters represent a type of phonologically constrained language that aims to engage a reader with high levels of phoneme overlap to encourage mispronunciations, often containing rhyme and humorous semantics, or simply conveying information in a form that is enjoyable to read due to the articulatory patterns that the lexical choices present. Consequently, tongue twister generation presents myriad unique challenges for NLG, including the need to consider the phonetic realization and underlying phonological representation of the chosen vocabulary, all while still maintaining a grammatically valid output sequence despite the often obscure and highly restricted candidate vocabulary. In addition to being a fruitful area for further investigation by the NLP/NLG communities, tongue twisters also present a wide range of real-world applications, making the case for their automatic generation even stronger. These applications include (1) being used as an educational tool for language teaching (Sugiharto, Santoso, and Shofyana 2022; Somoff 2014; Wilshire 1999); (2) being a source of entertainment and humor stemming primarily from unintentional misarticulation; (3) as a literary device for engaging young children in developing their literacy (such as the approach taken in numerous Dr. Seuss stories, Geisel 1965); (4) as a method of designing memorable slogans and tag lines (Guerini, Özbal, and Strapparava 2015); and (5) as stimuli in neuroscience and physiology research to investigate the localization of functions within the brain and how linguistic perception links to production on a neurological level (O'Halloran 2020; Wong et al. 2019; Kember, Connaghan, and Patel 2017). Consequently, the ability to automatically generate tongue twisters constrained on particular topics and phoneme combinations has many real-world applications. Moreover, findings from the generation of tongue twisters also have wider applicability in phoneme-conditioned language generation, such as the more widely studied areas

1 Code and resources available at `https://github.com/tylerL404/Train-and-Constrain-TT`.

of poetry and lyric generation, where being able to exert phoneme-level control of the output is desirable.

1.1 Contributions

Towards the automatic generation of high-quality tongue twisters, we expand upon prior work to present **TwisterLister**, an LLM-based pipeline for the generation of unique, non-derivative tongue twisters to provide more extensive training data to enable the training of smaller language models. TwisterLister employs semantic and phonological knowledge in the form of sentence embeddings and phonemic edit distance to restrict a candidate vocabulary list to pass to an LLM. In doing so, we create **TwistList 2.0**, the largest existing dataset of tongue twisters with over 17k examples, of which approximately 15k are derived directly from the proposed TwisterLister pipeline. We motivate the need for this extended wealth of training data by demonstrating the impact on both automatic metrics and human evaluation as a function of training-data volume by fine-tuning various smaller-scale language models (BART, Flan-T5, etc.) on various splits of this dataset. We present these results as part of two different tongue twister generation approaches, **topic-to-twister** and, inspired by Keh et al. (2023), **style-transfer** (i.e., prose-to-twister) (Figure 1). With the aforementioned real-world applications of tongue twisters, we motivate the topic-to-twister setting for applications such as language learning, where multiple outputs can be generated to test an individual's articulatory abilities in a new language while simultaneously expanding their vocabulary. On the other hand, the style-transfer setting is motivated by applications such as marketing, where a standard sentence conveying a desired meaning (e.g., a brand's mission statement or a product's features) can be reworded to have increased phonetic complexity to become a tongue twister, consequently engaging the reader and increasing memorability. We additionally introduce **PACD**, a **P**honeme **A**ware **C**onstrained **D**ecoding module, that can be used with any causal autoregressive language model to

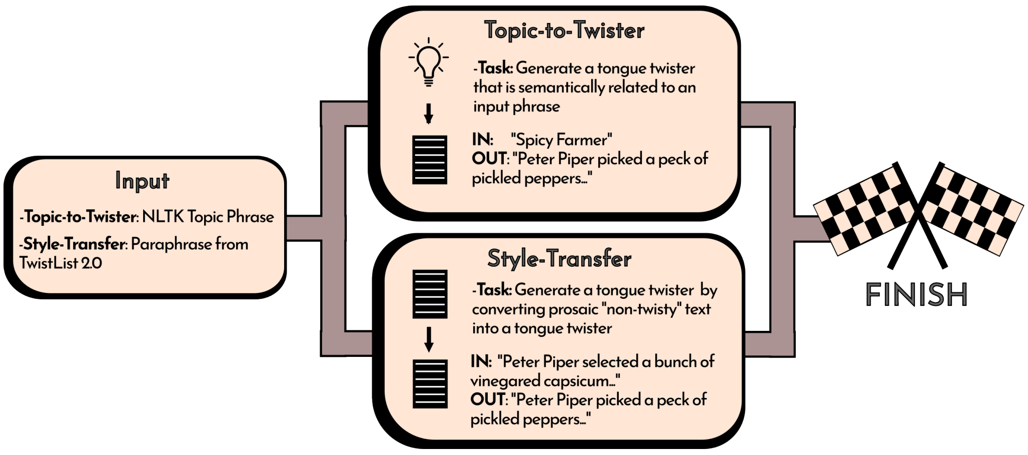

Input
-Topic-to-Twister: NLTK Topic Phrase
-Style-Transfer: Paraphrase from TwistList 2.0
Topic-to-Twister
-Task: Generate a tongue twister that is semantically related to an input phrase
IN: "Spicy Farmer"
OUT: "Peter Piper picked a peck of pickled peppers..."
Style-Transfer
-Task: Generate a tongue twister by converting prosaic "non-twisty" text into a tongue twister
IN: "Peter Piper selected a bunch of vinegared capsicum..."
OUT: "Peter Piper picked a peck of pickled peppers..."
FINISH

Figure 1
Outline of the tongue twister generation pipeline for both the topic-to-twister and style-transfer task settings. Both settings use the same base models and hyperparameters and differ only in the data used for fine-tuning.

ensure token outputs meet phoneme-level criteria. Overall, our contributions may be summarized as follows:

- **TwisterLister** - A phonologically and semantically informed pipeline for generating English tongue twisters with large language models, both as a stand-alone generation method and as a data synthesis approach for additional training data.
- **TwistList 2.0** - The most extensive English tongue twister dataset to date, containing 17,000+ tongue twisters produced via TwisterLister (∼15k) and human authors (∼2k), including extensive quality control procedures, for use in training tongue twister generation models as well as presenting a resource for the study of this language form on a linguistic level.[2]
- **PACD** - A phoneme-aware constrained decoding algorithm that applies hard lexical constraints on the outputs of autoregressive language models to achieve phoneme-level overlap and generate tongue twisters.
- **iPED/oPED** - Novel phonemic edit distance (PED) based metrics for assessing the articulatory characteristics of tongue twisters on a word-initial and overall basis.
- A range of experiments training smaller language models (i.e., GPT-2, DialoGPT, BART, Flan-T5, ByT5, and Baichuan) to generate tongue twisters in topic-to-twister and style-transfer settings using TwistList 2.0, including extensive automatic and human evaluation.
- Extensive qualitative linguistic analysis of generations from different models trained on varying quantities of training data, with or without the constrained decoding PACD module, in the form of case studies.

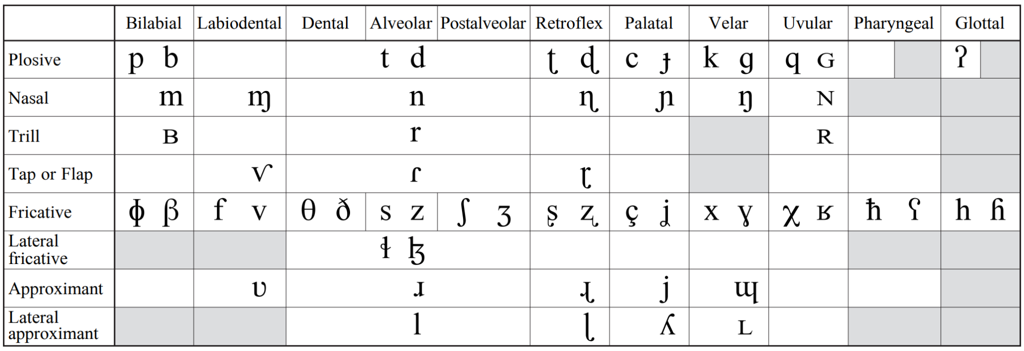
2. Phonetics and Phonology

Due to the strong reliance on theory and ideas from the fields of phonetics and phonology in this article, we find it apt to begin with a short introduction to these domains. Phonetics refers to an area of linguistics that studies the production and perception of speech sounds present in spoken languages (Gick, Wilson, and Derrick 2013; Jessen 2008; Ladefoged 1996) and the related field of phonology focuses more strongly on the abstract mental representations of speech sounds and the development of feature-based taxonomies for the categorization of related sounds (Clements and Ridouane 2011; De Lacy 2007; Klausenburger 1970).

2.1 Place and Manner of Articulation

Figure 2 presents the primary pulmonic consonants present in human languages. For each consonant, three important pieces of information on their **phonetic** characteristics

2 To the best of our knowledge, the previous record belongs to the original TwistList (1.0) from Loakman, Tang, and Lin (2023), containing just over 2.1k human-authored examples.



	Bilabial	Labiodental	Dental	Alveolar	Postalveolar	Retroflex	Palatal	Velar	Uvular	Pharyngeal	Glottal
Plosive	p b			t d		ʈ ɖ	c ɟ	k ɡ	q ɢ		ʔ
Nasal	m	ɱ		n		ɳ	ɲ	ŋ	ɴ		
Trill	ʙ			r					ʀ		
Tap or Flap		ⱱ		ɾ		ɽ					
Fricative	ɸ β	f v	θ ð	s z	ʃ ʒ	ʂ ʐ	ç ʝ	x ɣ	χ ʁ	ħ ʕ	h ɦ
Lateral fricative				ɬ ɮ							
Approximant		ʋ		ɹ		ɻ	j	ɰ			
Lateral approximant				l		ɭ	ʎ	ʟ			

Figure 2
The International Phonetic Alphabet consonant chart. Symbols to the right of a cell are voiced (i.e., involve the vibration of the vocal folds) and sounds to the left are unvoiced. Gray boxes present place/manner combinations that have been deemed impossible for human production. This chart is reproduced from the International Phonetic Association, which distributes it under the CC-BY-SA license.

are interpretable. First, the rows each represent a *manner* of articulation which refers to the physical process that occurs to produce a particular sound (such as a "plosive" involving the build-up and sudden release of air pressure within the mouth, whereas a "nasal" involves the nasal cavity through the lowering of the velum). Second, each column refers to a *place* of articulation, which relates to the main location that articulators (e.g., tongue, teeth, and lips) make contact within the vocal tract (e.g., "bilabial" sounds involve both lips and "labiodental" sounds involve the lips and teeth in their production). Finally, the last remaining detail is *voicing*, which refers to whether or not the glottal folds (also known as vocal folds or vocal cords) are vibrating during the sound's production.

2.2 Phonological Features

In addition to physical production-based characteristics, there are also further **phonological** features that can be used to explain the patterns and processes that phonemes undergo in human speech. An example of the different phonological features that phonemes may have is presented in Table 1. It is phonological features such as these that the phonemic edit distance (PED) measures introduced in §4 and used throughout

Table 1
A phonological feature chart. + means the feature is present for a given sound, whereas − means the feature is absent. Only a subset of possible sounds and phonological features are shown for demonstration purposes.

Sound	Consonantal	Sonorant	Voiced	Nasal	Continuant	Coronal
/t/	+	−	−	−	−	+
/d/	+	−	+	−	−	+
/s/	+	−	−	−	+	+
/l/	+	+	+	−	+	+
/i/	−	+	+	−	+	−

Table 2
Phonetic transcriptions of the phrase "Hello World" in the 3 most common standards. Note that this is just one of the possible productions. Accents differ in their phoneme inventories and phonological representations, which would necessitate different transcriptions.

Standard Orthography	IPA	ARPABET	SAMPA
Hello World	/hɛləʊ wəːld/	HH EH L OW W AX L D	hEl@U w@:ld

this article rely on to determine the similarity between two speech sounds (and therefore the likelihood of mispronunciation when transitioning between them).

2.3 Phonetic Transcription Standards

Multiple transcription standards exist for phonetic and phonological representations of text. In the majority of this article, the preferred standard is the International Phonetic Alphabet (IPA), where each speech sound is represented by a single symbol (e.g., /t, d, s, z/). The IPA is the standard transcription convention used within linguistic research and is used within Figure 2 and Table 1. We additionally make use of another transcription standard, ARPABET, which uses 1 or 2 characters to represent a given sound (or 3 where vowels have their stress marked). Table 2 presents an example of these standards, alongside another common standard, SAMPA.

3. Related Work

3.1 Creative Language Generation

Numerous efforts have been made toward the generation of creative language forms, with a range of findings regarding whether or not popular LLMs truly exhibit human-level creativity in these tasks. Chakrabarty et al. (2023) present work applying the Torrance Tests of Creative Thinking (TTCT) to objectively analyze the outputs of LLMs and human authors on a narrative writing task, finding that LLM generations perform measurably worse than humans, passing 3–10x fewer criteria outlined by the TTCT. However, it is important to note that this was in comparison to professional authors, who represent a very niche subset of the best human writers available. On the other hand, Gómez-Rodríguez and Williams (2023) compare human and LLM-authored narratives and find that LLMs are able to match or surpass human performance on several of the evaluation criteria they present. However, in this case, the "creativity" of the task was somewhat diminished by having prescribed rules about the topic, characters, and writing style, where the task may be more construed as emulating the writing of an existing work. However, similarly, both Franceschelli and Musolesi (2023) and Clark et al. (2021) observe that humans are infrequently able to distinguish creative works written by other humans from those authored by LLMs, with the latter often achieving very high-quality outputs. Overall, there is clear potential and room for improvement in the field of automatically generating creative language forms.

A popular trend is to investigate the extent to which models can be trained to generate language forms where training data is scarce. Wöckener et al. (2021) investigate this for the generation of poetry using ∼16k and ∼67k quatrains of English and German poetry, respectively, and notice difficulties in GPT-2 learning sub-lexical phenomena including rhyme from this number of training examples alone. However, poetry presents a highly restrictive form of literary language where many types contain formal constraints regarding length, syllable count, and metrical patterns. Additionally, Van de Cruys (2020) presents work on the generation of Shakespearean sonnets, another literary niche that contains an even more limited number of available training samples. They therefore approach the task via adding constraints at decoding time in a pipeline approach that includes the stages of content planning, rhyme generation, and output polishing to imbue literary sensibilities into the outputs of models trained exclusively on non-literary text. Finally, Popescu-Belis et al. (2023) use data synthesis techniques to increase the amount of rhyming data they can train GPT-2 (Radford et al. 2019) on, to realize GPoeT, which shows an increased ability to generate consecutive rhymes.

Tongue twister generation, as a niche subdomain of creative language generation (which in itself is a branch of NLG more generally), has only received attention in recent years. Keh et al. (2023) presented PANCETTA, the first major work on the automatic generation of this language form in the modern post-BERT (Devlin et al. 2019) NLP era, and released TT-Corp, a dataset of over 600 tongue twisters taken from various online sources. They train variations of naive and "phoneme aware" GPT-2-based models (Radford et al. 2019) in a topic-to-twister and style-transfer setting, progressing the inclusion of phoneme-level awareness by pre-training models on the IPA representation of WikiText data, and utilizing these models in conjunction with off-the-shelf orthographic models with the aim of exploiting the link between the phonological and orthographic representations of text. Shortly following PANCETTA, Loakman, Tang, and Lin (2023) presented the precursor to the present work and released TwistList (referred to as TwistList *1.0* in this article), a dataset of 2.1k human-authored tongue twisters collected from various online sources in line with Keh et al. (2023). Additionally, TwistList was used to train a wide range of language models, including GPT-2 (Radford et al. 2019), DialoGPT (Zhang et al. 2020b), BART (Lewis et al. 2020), and T5 (Raffel et al. 2020) solely in a topic-to-twister setting based on orthographic text. Two naive phonemic evaluation metrics were introduced, including Phoneme Overlap (PO) and Initial Phoneme Overlap (IPO), which assessed the homogeneity of the generations in terms of unique sounds. However, these metrics considered all sounds to be equidistant in phonological space (hence being naive).

Other forms of language where phonemic and phonetic information are essential have also been generated, including rap (Xue et al. 2021; Manjavacas, Kestemont, and Karsdorp 2019; Potash, Romanov, and Rumshisky 2018) and song lyrics more generally (Tian et al. 2023; Chang et al. 2023; Zhuo et al. 2023; Zhang et al. 2022). Computational research on creative works is not restricted to such domains, however, with extensive work also existing in the area of narrative generation (Hong et al. 2023; Tang et al. 2022; Chen et al. 2021), humor generation (Loakman, Maladry, and Lin 2023; Sun et al. 2022; Tian, Sheth, and Peng 2022), metaphor processing (Wang et al. 2024, 2023; Li et al. 2023a; Li, Guerin, and Lin 2022), and music generation (Li et al. 2024; Yuan et al. 2024; Yu et al. 2023).

3.2 Constrained Generation

An inherent paradox often exists when using language models for creative language generation. Models trained to generate the most probable sequence of tokens are used to output text where the most *probable* continuation is sometimes one of the least *preferable*.

In the simplest form, control over the output form can be exercised through simple methods such as restricting the output vocabulary at decoding time (Hokamp and Liu 2017; Valitutti et al. 2013), or through the application of penalties for *n*-gram repetition, therefore encouraging diversity (Zhang et al. 2021; Foster and White 2007). Several studies have been performed in the direction of toolkits to aid in constrained language generation. Roush et al. (2022) present the Constrained Text Generation Studio (CTGS), an artificial intelligence-writing assistant that in its most basic form applies a range of premade filters to the output of a probabilistic language model (such as "avoid the letter *e*", Wright 2016), scanning the most probable next-word generations and only selecting those that fall in line with the selected constraints (of which numerous can be applied in parallel). However, due to the design of the constraints, CTGS struggles with the generation of well-formed outputs for models trained using the predominant paradigm of sub-word tokenization. Similar approaches have also been utilized for the generation of other non-creative language types, such as machine translation and summarization, where stylistic constraints and editorial decisions may result in a preferred output structure. For example, Yao et al. (2023) present COLLIE, a grammar-based framework for the application of advanced compositional constraints on the outputs of language models, in addition to a tool for generating example task instances from raw text. Additionally, Iso (2022) presents AutoTemplate, a method of formatting a task structure to realize lexically constrained text generation. However, although many constraint-based systems exist, most work on lexical constraints is focused on the inclusion of specific word choices within the output, and therefore the desired candidate vocabulary has to be known a priori. Lu et al. (2022) propose to solve a common downfall of autoregressive decoding where it is necessary to plan ahead and introduce *NeuroLogic A*esque*, a decoding approach that uses lookahead heuristics to more carefully consider future token generations. We build upon work in the area of constrained generation in §7 with PACD, a phoneme aware constrained decoding approach that dynamically applies constraints on allowable tokens at each generation timestep.

3.3 Knowledge Distillation

Alongside the advent of LLMs, so too arose the area of knowledge distillation (Gupta and Agrawal 2022; Hinton, Vinyals, and Dean 2015; Buciluă, Caruana, and Niculescu-Mizil 2006) whereby the aim is to achieve the successful transfer of knowledge from a much larger model (often referred to as a teacher model) to one or more smaller models (often referred to as student models). In doing so, much more robust generalist models, such as GPT-3 (Brown et al. 2020) and GPT-4 (OpenAI et al. 2023), can have elements of their abilities passed down to smaller models that do not possess the same level of computing requirements (Yang et al. 2024). Although there are numerous methods of achieving the desired distillation, perhaps the simplest and easiest of these approaches, particularly in domains where data is scarce, is to generate new synthetic training data using the larger models, either via the generation of completely new examples or via data augmentation and perturbation (Whitehouse, Choudhury, and Aji 2023; Askari et al. 2023). However, generating novel instances is not always straightforward, with the quality of generations often being dependent on the task type. For instance, Li et al. (2023b) find that more subjective tasks may result in lower quality synthetic data, such as not reflecting the same level of diversity as human-written equivalents, something that is key to creative language domains and that we aim to overcome in §4.1 for the creation of TwistList 2.0.

4. TwistList 2.0 Dataset Construction

Loakman, Tang, and Lin (2023) presented *TwistList 1.0*, a dataset of 2.1k+ human-authored tongue twisters from various sources available on the Web including listicles and works of fiction. Although this allowed a high level of quality in the examples within the dataset (as all instances were automatically filtered, reviewed by a linguist, and then underwent quality control on a subset), the small size of this dataset (even while being the largest dataset of tongue twisters we are aware of to-date) means that smaller models struggle to learn the key features of tongue twisters from such limited examples, specifically regarding the need to balance high levels of phonemic repetition alongside maintaining grammatical coherence.

To combat this shortfall, we extend TwistList 1.0 8-fold into *TwistList 2.0*, containing 17,000+ unique tongue twister examples. To achieve this feat, we note the near-human performance that was achieved by early versions of ChatGPT in our previous human evaluation studies (Loakman, Tang, and Lin 2023), and opt to build a generation pipeline we name *TwisterLister* using GPT-3.5-Turbo to generate novel examples to facilitate the training of smaller parameter count models from the resulting synthetic dataset.[3]

4.1 TwisterLister Pipeline

A key discovery in previous work (Loakman, Tang, and Lin 2023) was the common reliance of ChatGPT on slightly modifying well-known existing tongue twisters that had been memorized from the training data when presented with a new topic (for example, "*silver shiny ships*" generated "*How much wood could a woodchuck chuck if a woodchuck could chuck silver shiny ships*"). To avoid this pitfall, we create a more constrained pipeline for tongue twister generation that promotes the generation of unique, non-derivative examples, illustrated in Figure 3. Initially, we generate the topics for the tongue twisters by building a set of topic phrases by combining a randomly sampled adverb or adjective with a noun to represent an abstract topic (using part-of-speech tags from Natural Language Toolkit's (NLTK)'s Brown Corpus [Francis and Kucera 1979]). Following this, we then randomly select a phoneme on which to focus the tongue twister, restricting these choices to consonants (due to these being more commonly exploited in tongue twisters than vowels) and additionally removing any phonemes that are not legal in word-initial position in standard English phonotactics (such as the glottal stop /ʔ/) or phonemes with very few entries in our vocab bank (such as the voiced postalveolar fricative /ʒ/). Following this, we search the CMU Pronouncing Dictionary (CMUDict)[4] for words starting with our preferred phoneme and calculate the cosine similarity between the SentenceBERT embedding of our NLTK-generated topic phrase with each candidate word retrieved from the CMUDict. Following this, the top N retrieved candidate words with the highest semantic similarity are kept, and the others are discarded (N = 5 or 10).

We then calculate the pairwise weighted phonemic edit distance (PED) between our initially selected phoneme and all other allowable phonemes, and select the lowest scoring phoneme (i.e., most similar) to act as the best secondary phoneme.[5] Where

3 In both Loakman, Tang, and Lin (2023) and the present work, we access GPT-3.5-Turbo (i.e., "ChatGPT") via the OpenAI API.
4 Available at `https://github.com/Alexir/CMUdict/tree/master`.
5 Phonemic edit distance is implemented with the *panphon* package (Mortensen et al. 2016).

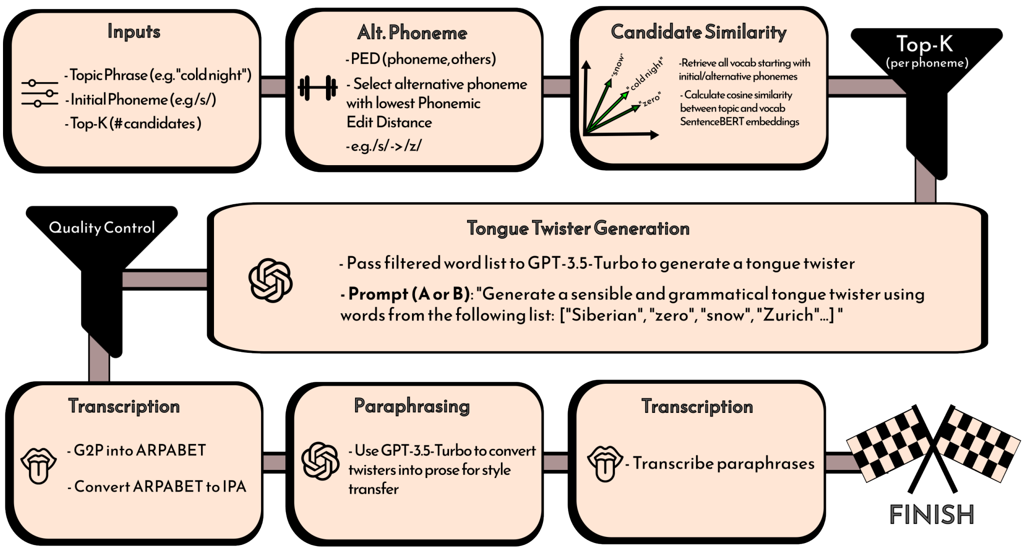

Inputs
- Topic Phrase (e.g. "cold night")
- Initial Phoneme (e.g /s/)
- Top-K (# candidates)
Alt. Phoneme
- PED (phoneme, others)
- Select alternative phoneme with lowest Phonemic Edit Distance
- e.g. /s/ -> /z/
Candidate Similarity
"snow"
"cold night"
"zero"
-Retrieve all vocab starting with initial/alternative phonemes
-Calculate cosine similarity between topic and vocab SentenceBERT embeddings
Top-K (per phoneme)
Quality Control
Tongue Twister Generation
- Pass filtered word list to GPT-3.5-Turbo to generate a tongue twister
- Prompt (A or B): "Generate a sensible and grammatical tongue twister using words from the following list: ["Siberian", "zero", "snow", "Zurich"...] "
Transcription
- G2P into ARPABET
- Convert ARPABET to IPA
Paraphrasing
- Use GPT-3.5-Turbo to convert twisters into prose for style transfer
Transcription
- Transcribe paraphrases
FINISH

Figure 3
Illustrated outline of the TwisterLister dataset generation pipeline used to create TwistList 2.0. PED refers to phonemic edit distance and G2P refers to grapheme-to-phoneme conversion.

multiple share the same edit distance, the first reached when iterating over the list is selected. This is due to tongue twisters frequently relying on the reader mispronouncing a sound, often due to confusion with a phonetically/phonemically similar sound (e.g., “she sells sea shells” exploiting /ʃ/ and /s/). Consequently, allowing the generation process to select from 2 banks of words that start with similar sounds means that we can more directly promote mispronunciations rather than solely relying on the repetition of a single sound. We then repeat the process of generating a candidate list of words that are semantically related to the topic and start with our desired secondary phoneme.

We then combine the word lists and shuffle their order to promote alternation between words with different, yet similar, initial phonemes as experimentation showed that requesting the LLM to “alternate” between words from 2 different word banks, and presenting them separately, still resulted in the words being used largely in the same order that they were presented, and therefore not resulting in the desired alternation. The list is then fed into the LLM (GPT-3.5-Turbo, accessed August-September 2023 via the API) with one of the prompts presented in Table 3.

In Prompt A, where we specify single-sentence outputs, we see much more concise outputs that may suffer from coherence issues, whereas in Prompt B we see more coherent outputs but which often resemble standard poetry more than tongue twisters (to which we apply further filtering as discussed in §4.3). Consequently, we use a combination of tongue twisters generated with either prompt to have a diverse range of styles.[6] All generations are performed with *max_tokens* set to 1,000 and a temperature

6 We did not perform extensive prompt-engineering to arrive at these but observed the differing behavior in our preliminary testing.

Table 3
The two prompts used in the TwisterLister pipeline for tongue twister generation from GPT-3.5-Turbo. *[word-list]* refers to the phonetically and semantically conditioned vocabulary selected in the previous steps.

Prompt A
"Generate a sensible and grammatical tongue twister using words from the following list: [word-list]. The output should be a single sentence and be grammatical and coherent."
Prompt B
"Generate a tongue twister by primarily using words from the following list: [word-list]. The output should be grammatical and coherent."

of 0.8 to facilitate creative completions. In total, we generated 17,500 example tongue twisters, of which 11,500 were generated with Prompt A, and 6,500 with Prompt B.[7]

4.2 Style-transfer Paraphrase Generation

In line with Keh et al. (2023), we present an additional task setting alongside the topic-to-twister approach that utilizes style-transfer. Whereas Keh et al. (2023) generate non-tongue twister examples of their dataset entries via simple rule-based synonym replacement conditioned on part-of-speech tags, we leverage GPT-3.5-Turbo to paraphrase each entry in our tongue twister dataset. To achieve this, we pass the following system prompt to GPT-3.5-Turbo: *"In this task you will pretend that you're an author who is rewriting existing works into a non-literary form that more resembles prose. You will be presented with a tongue twister and asked to rewrite it using synonym replacement so that there are no longer high levels of phonetic overlap and sound repetition. Example 1: INPUT = "She sells sea shells by the seashore." OUTPUT = "The girl sells conches by the ocean." Example 2: INPUT = "Peter Piper picks pickled peppers" OUTPUT = "Peter Piper selects preserved capsicums"*. We then present the following *user* message to GPT-3.5-Turbo for each dataset entry: *"INPUT = "[twister], OUTPUT = "* where *[twister]* is a standard tongue twister from TwistList 2.0. This approach is superior to simple synonym substitution as the LLM can dynamically select new vocabulary terms in a way that further ensures the new paraphrase uses combinations of vocabulary that are sensible. This is because raw synonym replacement can result in semantic drift (as very few words are true synonyms), potentially risking nonsensical outputs (Chiang and Lee 2023).

4.3 Refining Outputs

To further refine the generated outputs, we process the resulting dataset in several stages. First, all outputs generated using Prompt A (which promotes succinct tongue twisters that are not as coherent) are re-fed into GPT-3.5-Turbo in the prompt *"Improve the following tongue twister by editing it so that it makes more sense and is grammatical: [tongue*

7 The imbalance here is due to wishing to promote the generation of "twisty" content, which is more prevalent with Prompt A. Consequently, we sample fewer Prompt B examples to contain a moderate amount of literary/poetic text.

twister]". Consequently, this step fixes errors that arise from the original word lists given to the initial prompt containing morphological variants that are difficult to turn into a coherent output (for example, all nouns being in possessive form, or verb tenses being mixed in a way that is detrimental to coherence). To further check that the remaining outputs are sensible, we calculate the perplexity (PPL) of the generations using a pre-trained language model (in this case GPT-2) as a basic heuristic for well-formedness. We then compare these scores with the average PPL across the original TwistList dataset of human-authored tongue twisters and remove any outputs from our new dataset that have perplexities that are higher than the original dataset's mean plus one standard deviation. This stage removed 397 examples.

Additionally, as previously mentioned, Prompt B promoted the generation of longer, more poetic outputs, at the cost of not always resembling tongue twisters. In order to maintain only the best generations, we apply a metric for assessing the phonological characteristics of tongue twisters using weighted PED (Mortensen et al. 2016), which is outlined in further detail in §5.3. Again, we compare the results to the mean score from the original TwistList dataset, and filter our new dataset by removing any examples that do not score lower (i.e., better) than the original dataset's mean plus one standard deviation. However, no examples were caught in this filtering stage, suggesting that the majority of the more poetic works still exhibited tongue-twister-esque phonetics.

Next, to encourage diversity, we remove examples based on word-overlap. To achieve this, we apply a pairwise fuzzy-matching algorithm based on the token sort ratio (which is indifferent to word order) across the remaining tongue twisters and remove examples with more than a 60% overlap with an existing twister in the dataset.[8] This step consequently led to the removal of 1,747 entries, therefore reducing repetition and increasing diversity in the remaining dataset.

We then filter out offensive examples that may have been created by poor topic/phoneme combinations that resulted in undesirable stereotypes or associations being expressed. These include words relating to the topics of racism, sexism, homophobia, transphobia, and additional terms that some may find offensive (including general expletives and references to various anatomy). We perform this by comparing the tongue twisters with a bank of offensive words and removing any twisters containing any of the examples (regardless of context). This stage removed a final 69 entries from our dataset.

Finally, to ensure the diversity of input/output pairs when training in a topic-to-twister fashion, we remove any entries with duplicate NLTK topic phrases, as these cases would result in 2 tongue twister outputs for a single input (but with different phonological characteristics, as they would have been caught and filtered by previous stages if not). This stage removed a final 135 examples.

The final additions to the dataset comprise 15,151 examples, which when combined with the existing TwistList 1.0 results in 17,278 unique tongue twisters in TwistList 2.0. We maintain a distinction between human- and machine-authored tongue twisters in the final dataset so that different communities can make use of whichever is most pertinent to their application.

8 We arbitrarily decide which to remove and continue until no remaining samples exhibit this level of overlap due to these examples having passed the other two layers of filtering, and therefore being acceptable. We implement this using the RapidFuzz package: `https://github.com/rapidfuzz/RapidFuzz`.

4.4 Additional Processing and Quality Control

As in Loakman, Tang, and Lin (2023), we then enhance the dataset with the addition of phonetic transcription using the *g2p-en* Python package.[9] We experimented with other grapheme-to-phoneme (G2P) solutions to see if an improvement was possible here such as SoundChoice (Ploujnikov and Ravanelli 2022) which better accounts for correctly transcribing homographs, but opted to utilize g2p-en again. This is primarily due to our dataset generation pipeline relying on the CMUDict to retrieve vocabulary, and g2p-en queries the CMUDict for transcriptions before falling back on a trained neural network model to infer pronunciations for out-of-vocabulary tokens. Consequently, the majority of our less common vocabulary will already have a gold standard transcription in CMUDict (in the General American accent, due to the limitations of this resource). We then convert these transcriptions into the IPA to facilitate the use of our phoneme-based metrics, but also to provide a transcription standard that is more common in the linguistics domain.

Finally, whereas in Loakman, Tang, and Lin (2023) RAKE (Rose et al. 2010) was used to extract keywords from the human-authored examples to represent the topic, we skip this stage here and utilize the topic phrases we used in the dataset generation step. As a result, unlike the keywords for the original tongue twister collection, those that are new to TwistList 2.0 have more abstract topics as the topic words are not forced to appear in the output twister (rather, only a semantic link is present). We hypothesize that this may also help to reduce drawbacks seen in the original work, where our trained models often repeated the topic keywords numerous times to achieve a "tongue twister", rather than learning a deeper representation of semantics.

4.4.1 Quality Control. Quality control on our dataset was performed in multiple ways. Five human evaluators, who are native speakers of English, were provided with 50 sampled instances from the dataset from different conditions to rate the quality of the resulting tongue twisters (25 from Prompt A and 25 from Prompt B). Scores for the criteria were given on a scale from 1 (low quality) to 5 (excellent quality) for 5 criteria: (i) "Twister" refers to the assessed quality of the tongue twister, and whether it exhibits the expected characteristics of a traditional tongue twister (and is analogous to the "overall" score present in later human evaluation of our model generations in §6.1). (ii) "Topic" refers to how well the input topic phrase is represented in the output via semantics. (iii) "Paraphrase Quality" refers to whether or not the paraphrase generation step maintains a meaningful and grammatical text, and (iv) "Paraphrase Prosaic" refers to what extent the paraphrase is believed to have successfully removed the sound overlap and tongue twister nature of the original input to more resemble standard text. Finally, (v) "Overall" offers a holistic assessment of the dataset entry as a whole. Table 4 presents the breakdown of the human evaluation results.

In sum, what may be concluded is that the resulting dataset is considered to be of good quality, particularly with regard to the paraphrased versions and overall evaluation. When compared to other aspects, the ratings for the tongue twisters themselves and topic semantics are lower but still indicate reasonably good quality (with 3 being the middle rating, akin to "neither agree nor disagree"). This is particularly true when considering that tongue twister quality is a highly subjective measure due to entertainment value being a fundamental component that different people perceive to different

9 Available at: `https://github.com/Kyubyong/g2p/tree/master`.

Table 4
Results of human quality control assessments for TwistList 2.0. Importantly, we only sample from the new machine-generated samples created with TwisterLister for quality control. Assessments regarding the additional 2k+ human-authored examples can be found in the original TwistList work (Loakman, Tang, and Lin 2023), where 88% of evaluated tongue twisters were deemed "high quality" (scoring 3 on a 3-point scale). Furthermore, the "Golden" samples evaluated in §6.2 are exclusively from the human-authored portion of TwistList 2.0, due to these constituting the test set. We calculate Fleiss' Kappa for each metric and mark the agreement fair*, moderate**, and substantial***.

TwistList 2.0 Quality Control				
Twister	Topic	Paraphrase Quality	Paraphrase Prosaic	Overall
3.138**	3.333*	4.421*	3.969**	3.778**

levels. Additionally, the topic association is lower due to not enforcing that the input terms are present in the output. On the other hand, extracting keywords from twisters and using these as the inputs would lead to an artificially inflated rating for the topic criteria, as the topic would be guaranteed to be represented explicitly in the output. Overall, the dataset samples were given a mean rating of 3.778, akin to a rating very close to "high quality", which is rather good for a creative, and therefore subjective, domain. Further details of human participant recruitment are reported in §5.4.

4.5 TwistList 2.0 Dataset Summary

Statistical details of TwistList 2.0 can be seen in Table 5. Additionally, two example entries in the dataset are outlined in Table 6. As the examples demonstrate, the combination of adjective/adverb and noun used as a topic phrase during the TwistLister generation pipeline can be directly output as part of the final twister, or can instead be represented solely by semantics. For example, in the top example (TT_ID 68), the input adjective "public" is replicated in the output. The reason for this is due to the chance of the phoneme /p/ being selected at random at generation time. Consequently, it makes

Table 5
The statistics of TwistList 2.0. The validation set is obtained by randomly selecting the same volume of data as the test set from the shuffled training set. All of the test set input phrases were extracted from TwistList *1.0* entries using RAKE (Rose et al. 2010), rather than being topics we generated using part-of-speech tags. We do this in order to test our models on more robust input forms (where input length is not static), therefore facilitating more possible input combinations than adjective/adverb + noun would allow. This additionally allows comparison to human-authored gold standards in our later testing. *Train (A)* and *Train (B)* refer to samples generated with Prompt A and Prompt B, respectively.

Dataset	**Train (A)**	**Train (B)**	**Full Train**	**Test**	**Total**
# Tongue Twisters	9,653	5,471	15,124	2,124	17,248
# Vocabulary Size	60,306	61,160	93,517	10,336	98,703
Avg. # Topic Words	2.00	2.00	2.00	3.16	2.14
Avg. # Paraphrase Words	26.63	57.51	37.80	17.19	35.26
Avg. # Tongue Twister Words	23.90	53.60	34.64	14.98	32.22

Table 6
Example entries from TwistList 2.0. All entries also contain IPA transcriptions used by converting ARPABET directly into IPA. Consequently, some minor pronunciation errors occur as a result of an imperfect mapping between the two standards and the selected accent of the G2P tools used. We forego presenting the transcriptions for the second example in the interest of space. In the dataset, phonemes that constitute a single word in ARPABET are separated with a space, and separate words are delineated via double spaces. IPA transcriptions contain no separators within words (due to being a 1:1 symbol:sound mapping), with a single space between words.

TT ID:	68
Topic:	"public commentator"
Source:	GPT-3.5-Turbo
Prompt:	A
Tongue Twister:	The public-spirited BBC broadcasts presented by persistent presenters perplexed the publicist Berman the broadcaster and the pollster profoundly.
Paraphrase:	The civic-minded BBC airs hosted by determined hosts confused the publicist Berman the broadcaster and the poller deeply.
Twister ARPABET:	DH AH0 P AH1 B L IH0 K S P IH1 R IH0 T AH0 D B IY2 B IY0 S IY1 B R AO1 D K AE2 S T S P R IY0 Z EH1 N T AH0 D B AY1 P ER0 S IH1 S T AH0 N T P R EH1 Z AH0 N T ER0 Z P ER0 P L EH1 K S T DH AH0 P AH1 B L IH0 S IH0 S T B ER1 M AH0 N DH AH0 B R AO1 D K AE2 S T ER0 AH0 N D DH AH0 P OW1 L S T ER0 P R OW0 F AW1 N D L IY0.
Paraphrase ARPABET:	DH AH0 S IH1 V Z IH0 K EY2 D M IH0 D B IY2 B IY0 S IY1 EH1 R Z HH OW1 S T IH0 D B AY1 D IH0 T ER1 M AH0 N D HH OW1 S T S K AH0 N F Y UW1 Z D DH AH0 P AH1 B L IH0 S IH0 S T B ER1 M AH0 N DH AH0 B R AO1 D K AE2 S T ER0 AH0 N D DH AH0 P AA1 L ER0 D IY1 P L IY0.
TT ID:	10397
Topic:	"direct language"
Source:	GPT-3.5-Turbo
Prompt:	B
Tongue Twister:	Non-direct, multilingual, and non-verbal, Monolingual speakers were nominal. Some could speak Mandarin natively, While others relied on mediated means. The novelistic twist began when mispronouncing, Novellus words of complex morphology. Non-native speakers struggled with Macromedia, And nonnatives found Mandarin challenging. Nonaccrual in Marathi, they preferred, As net learners of a morphological world. But despite the hurdles, they persevered, Finding beauty in languages unfurled.
Paraphrase:	Indirect, polyglot, and non-verbal, Single-language speakers were nominal. Some could converse in Mandarin from birth, While others depended on mediated methods. The narrative turn commenced when mispronouncing, Novellus terms of intricate structure. Non-indigenous speakers struggled with Macromedia, And non-natives found Mandarin demanding. Non-accumulation in Marathi, they favored, As online learners of a structural world. But despite the obstacles, they persisted, Discovering elegance in languages unfolded.

sense to expect "public" (and morphological variants thereof), to be in the top-k most semantically relevant words to the topic word "public", and therefore constitute part of the constrained candidate vocabulary. The clear selection of the phoneme /b/ as the secondary phoneme (selected via minimizing phonemic edit distance) can also be seen, where words such as "broadcaster" and the proper noun "Berman" have been selected (where /p/ and /b/ differ only in the presence/absence of voicing). Similarly, in the second example (generated from the input "direct language") we see "non-direct" in the output due to the selected initial or secondary phoneme being /n/ or /m/ (alveolar and bilabial nasal consonants, respectively), and "language" has been referenced less directly via terms such as "monolingual", "novelistic", and "Mandarin". The impact of the different prompt forms can also be seen, with the first example (using Prompt A) being much more succinct than the second example (which used Prompt B). Finally, regarding the paraphrasing used to enable training of style-transfer models, it can be seen that GPT-3.5-Turbo maintains the grammaticality and coherence of the original tongue twister, but replaces much of the vocabulary. However, some terms remain for semantic reasons, such as “BBC” in the first example and "non-verbal" in the second example.

Due to exercising limited control over the paraphrasing stage, some synonym replacements may result in maintaining similar levels of sound overlap (at least in the word-initial alliterative sense). For instance, “pollster” has been replaced by “poller” in the first example, and “methods” has been replaced by “means” in the bottom example. The reason for the former example is similar to why the input phrase *may* be in the output of the final tongue twister as the closest synonym to many words is a derivative of the word itself (i.e., variants of “poll-”).

5. Topic-to-Twister and Style-Transfer Tongue Twister Generation

5.1 Task Definition

Inspired by Keh et al. (2023), we define two different settings for the task of tongue twister generation we describe as *topic-to-twister* and *style-transfer*. In the former topic-to-twister setting, for a given topic phrase (as generated by randomly sampling adjective and noun combinations, as described in §4.1), we aim to generate a tongue twister T, whereby T comprises a sequence of words $\{w_1, w_2, \dots w_n\}$. The generated output must satisfy the following constraints: (1) the output should be semantically related to the input topic phrase; (2) the output should show maximal levels of phonological overlap across tokens; and (3) the output should be grammatically valid. On the other hand, for the style-transfer setting we aim to generate a tongue twister again, $T = \{w_1, w_2, \dots w_n\}$, but provide as input a non-tongue twister phrase and aim to convert it via style-transfer into a tongue twister through learning to replace vocabulary with more phonemically similar entries.

5.2 Trained Models

In order to realize our goals of tongue twister generation in topic-to-twister and style-transfer settings, we fine-tune a range of popular language models with varying parameter counts on our TwistList 2.0 dataset.

- **GPT-2** (117M) (Radford et al. 2019) - A popular transformer-based text generation model consisting of both an encoder and a decoder.
- **DialoGPT** (117M) (Zhang et al. 2020b) - A version of GPT-2 that has been fine-tuned on an extensive corpus of dialogues to enable better conversational performance (and therefore consequently often better understands natural language task prompts).
- **BART** (139M) (Lewis et al. 2020) - A popular denoising autoencoder model consisting of a BERT-like encoder (Devlin et al. 2019) with a GPT-2-like decoder (Radford et al. 2019).
- **Flan-T5** (250M) (Chung et al. 2022) - A further instruction fine-tuned version of the T5 model (Raffel et al. 2020).
- **ByT5** (582M) (Xue et al. 2022) - A version of T5 trained with byte/character level tokenization, rather than subwords.

- **Baichuan** (7B) (Yang et al. 2023) - A large-scale open-source LLM trained on English and Mandarin, achieving SoTA performance on many tasks for a model of its size. Due to the significant size of Baichuan, we perform fine-tuning with the help of Low-Rank Adaptation (LoRA) (Hu et al. 2022).
- **ChatGPT** (GPT-3.5-Turbo) (Ouyang et al. 2022) - A large language model fine-tuned for chat-based interactions and instruction following, which excels in few and zero-shot tasks. Importantly, we use ChatGPT in a zero-shot manner and do not perform any fine-tuning.[10]

5.2.1 Training Splits. In order to investigate the benefit of access to different amounts of training data for a data-driven approach to tongue twister generation in topic-to-twister and style-transfer task settings (and to motivate the contribution of our large dataset), we train all of the above models on training sets of various sizes, including 2k, 4k, 8k, and 13k training samples from TwistList 2.0. One exception here is Baichuan, which due to computation requirements we train only on the largest 13k split. We keep the test set (for automatic evaluation) to 2,124 samples, covering the entirety of TwistList 1.0, and have a validation set of equal size. Due to using TwistList 1.0 entries as our test set, all reference-based metrics are compared to human-authored outputs, and human evaluation scores can be directly compared to human performance.

5.2.2 Hyperparameters and Training Details. To leverage pre-trained parameters, we restore the encoder, decoder, and embedding layers from public checkpoints: (1) **GPT-2**: gpt2-base `https://huggingface.co/gpt2`; (2) **DialoGPT**: DialoGPT-medium `https://huggingface.co/microsoft/DialoGPT-medium`; (3) **BART**: bart-base `https://huggingface.co/facebook/bart-base`; (4) **Flan-T5**: flan-t5-base `https://huggingface.co/google/flan-t5-base`; (5) **ByT5**: byt5-base `https://huggingface.co/google/byt5-base`. The checkpoints of Baichuan are the exception, which require downloading the weights directly from their repositories (`https://github.com/baichuan-inc/Baichuan2`). Here we use the 7B checkpoint of Baichuan 2. As the vanilla Baichuan model requires extremely large computing resources, we implement the LoRA (Hu et al. 2021) technique to reduce computational costs. LoRA adapts large language models by incorporating low-rank modifications. This approach involves adjusting or fine-tuning extensive language models to address specific tasks or domains with a reduced computational cost. By introducing low-rank modifications, LoRA aims to enhance the adaptability and efficiency of these models while maintaining their performance. This technique is particularly beneficial for tailoring pre-trained language models to better suit specialized or narrower domains without requiring excessive computational resources. Our use of ChatGPT consists of GPT-3.5-Turbo via the OpenAI API, where we provide no information other than the same prompt we use with all other models. All other settings remain at default.

Our local experiments are carried out on a single Nvidia A40 GPU, which has 48GB of VRAM. When training neural models, we implement the PyTorch Lightning framework to set up training processes. The training parameters are as follows: The *batch size* is set to 16 (excl. Baichuan w/LoRA, where it is 8); the *learning rate* is 1e-4; *max*

10 GPT-3.5-Turbo was accessed for this purpose during August-September 2023 via the Chat Completions API.

source length is set to 512, and the *max target length* is set to 100; the optimizer uses Adam (Kingma and Ba 2014), and the ϵ of Adam is set to 1e-8. The whole training process lasts for 10 *epochs*, and the validation checking runs every half epoch. However, the presented results only consider the checkpoint with the best performance (i.e., lowest loss).

For all topic-to-twister settings, we use the prompt *"Generate a tongue twister on the topic of "[TOPIC]""* where *[TOPIC]* refers to the input topic phrase from the test set, and for all style-transfer settings we use the prompt *"Generate a tongue twister by rewriting the following text: [PARAPHRASE]"* where *[PARAPHRASE]* refers to the non-literary paraphrase of a tongue twister from the test set.

5.3 Automatic Metric Suite

We present extensive automatic evaluation on the following metrics: **Perplexity** (**PPL**), **BLEU** (**B-1/B-2/B-3/B-4**) (Papineni et al. 2002), **ROUGE** (**Ro-1/Ro-2/Ro-L**) (Lin 2004), and **BERTScore** Precision, Recall, and F-Measure (Zhang et al. 2020a) (**BS-P/BS-R/BS-F**). PPL, BLEU, and ROUGE are standard metrics in language generation to assess quality, and BERTScore assesses semantic similarity to a gold reference. It should be noted that due to the nature of our task, many potential "gold standard" tongue twisters exist for any given input. Consequently, as with all creative generation works, these reference-based metrics should be interpreted cautiously while being aware of their limitations (although we opt to include them for completeness).

5.3.1 Readability. Tongue twisters are known for their intricate phoneme-level patterns and linguistic complexity, making them challenging to articulate correctly. Consequently, readability metrics can be used to indirectly measure whether or not tongue twisters, due to their complex nature, are using more complex vocabulary in order to meet strict phonemic constraints (such as when a selected phoneme only has a few obscure words that are related to the input topic). We present a range of readability metrics, incorporating the **Dale-Chall Readability Index** (Chall and Dale 1995) (**Re-D**), **Flesch–Kincaid Readability Score** (Flesch 1948) (**Re-F**), **Gunning-Fog Index** (Gunning 1971) (**Re-G**), and **ARI Index** (Smith and Senter 1967) (**Re-A**). These comprise a series of complexity and readability metrics that relate to the necessary comprehension level of a text's audience. They calculate a numerical score based on factors such as sentence length and word complexity, offering a quantitative measure of how difficult a text is to understand. These further complement our other metrics by analyzing linguistic complexity via means other than phonology.[11] Additionally, extremely high scores on such metrics can likewise indicate the nonsensical nature of any particularly poor generations. This is because readability metrics take into account factors such as sentence length and syllable counts, with high scores being given to indefinitely long sequences (due to the absence of sentence-final punctuation) as well as convoluted and overly complex syntactic structures and lexical choices.

5.3.2 Phonology/Phonetics. We further develop our tongue twister measures and PEDs relying on the *weighted phonemic edit distance* function from the *PanPhon* package (Mortensen et al. 2016). This allows us to more directly analyze the phonetics of our generated outputs by taking into consideration the articulatory similarities and differences

11 All readability metrics were implemented from `https://pypi.org/project/py-readability-metrics/`.

between different phonemes. For example, previous phonetic metrics from Loakman, Tang, and Lin (2023), PO and Init-PO, treat all phonemes as equidistant in feature space, resulting in a transition from /s/ to /ʃ/ being viewed as the same "quality" as a transition from /s/ to /g/. In the former case, we have transitioned from a voiceless alveolar fricative to a voiceless *post*-alveolar fricative (therefore moving the tongue slightly further back in the mouth), whereas in the latter case the transition is between a voiceless alveolar fricative and a voiced velar plosive, resulting in a change of voicing, position, and manner of articulation. Consequently, using phonemic edit distance allows us to not punish transitions between phonemically similar sounds (which are more likely to encourage mispronunciations, such as in "she sells sea shells") as heavily as we punish transitions between unrelated sounds. As with PO and Init-PO, we utilize this weighted edit distance on both a word-initial and overall level, with **oPED** taking the *overall* average edit distance between every phoneme transition in the tongue twister, whereas **iPED** calculates the edit distance between word-*initial* phonemes (and is, therefore, a more accurate measure of "soft" alliteration).

Formally, for iPED and oPED let X and Y be two phonemes, represented as feature vectors, where each vector contains binary values representing the presence or absence of a phonological feature. Therefore, the weighted feature edit distance between X and Y is defined as: $D_{\text{wfe}}(X, Y) = \min_{\text{alignments}} \sum_{i=1}^{n} w_i \cdot d_i$, where X and Y are sequences of phonological features represented as feature vectors, n is the length of the longest common subsequence of X and Y, d_i represents the distance between the i-th feature in the alignment, and w_i is the weight assigned to the i-th feature. The minimum is taken over all possible alignments of the sequences X and Y. Each alignment assigns a distance d_i between corresponding features in the sequences. The weighted sum of these distances is computed, where each distance is multiplied by its corresponding weight w_i. For iPED, we calculate the mean distance across sequences of word-initial phonemes, whereas for oPED we calculate the mean distance when comparing all adjacent phonemes across the tongue twister. This metric provides a flexible way to measure the similarity between sequences of phonological features, allowing for customization through feature weighting. In the case of oPED and iPED, the lower the score, the better (with a score of 0 relating to 100% overlap of a single phoneme throughout the tongue twister).

In addition to our novel metrics discussed above (iPED/oPED), we additionally present our two metrics from Loakman, Tang, and Lin (2023), Phoneme Overlap (**PO**) and Initial Phoneme Overlap (**Init-PO**). **PO** refers to the average overlap of all phonemes across tokens (# unique phonemes / # total phonemes), whereas **Init-PO** is the ratio of unique word-initial phonemes to the number of words (# unique word-initial phonemes / # words). Although these original phoneme-based metrics reward longer outputs, we argue that all other things equal, a longer tongue twister is better than a shorter one as it provides more entertainment and more opportunities for mispronunciation. Perfect scores on PO and Init-PO can be achieved by the repetition of a single word. Although this does not lead to high-quality outputs, these metrics are intended exclusively to be indicators of sound characteristics, rather than an overall guide to quality. In both cases, higher levels of overlap result in lower ("better") scores, and the highest ("worst") achievable score is 1.

5.4 Human Evaluation Protocol

Due to the limitations of automatic evaluation metrics and tongue twisters being a creative domain where articulation abilities are tested, we also perform human evaluation. In line with Loakman, Maladry, and Lin (2023), we aim to be transparent in our human evaluation of a subjective language type that may be considered a type of humorous language. In total, five evaluators were asked to rate 20 outputs from the best performing standard baselines, Flan-T5 and ByT5, in addition to Baichuan, ChatGPT (i.e., GPT-3.5-Turbo), and golden examples from **TwistList 2.0**, on the following criteria: **Relevance** (how relevant the tongue twister is given the keyword inputs), **Fluency** (how grammatically valid the output is), **Difficulty of Articulation** (how difficult a tongue twister is to say), **Coherence** (how much sense the output makes), and **Entertainment Value** (how entertaining the output is, considering sounds and semantics) and a holistic **Overall** criteria. All ratings were given on a 5-point scale where 1 equates to "poor" and 5 equates to "excellent". Importantly, we include both Flan-T5 and ByT5 trained on 2k samples, as well as 13k samples, to investigate whether or not increased training data has a measurable impact on human evaluation in addition to the patterns observed in automatic evaluation.

5.4.1 Evaluator Recruitment and Demographics. In total, we recruit 5 evaluators via internal notices and word of mouth. All evaluators have university-level education to a minimum of undergraduate level in a range of fields including linguistics, computer science, engineering, and animation, and therefore represent a wide range of academic backgrounds. All evaluators are native speakers of English and report no language processing issues. Evaluators were provided with a 55GBP Amazon gift card for their combined work on dataset quality control and output evaluation, totalling approximately 4 hours of work.

5.4.2 Materials Provided to Evaluators. Evaluation was performed on an online platform. Participants were provided with a page of detailed instructions on how to navigate the platform and the order in which to perform evaluation tasks. Importantly, all human evaluation responses were given on a 1–5 rating scale (where 1 equates to "poor" and 5 equates to "excellent" for how well a criterion is met). For example, "*The tongue twister can be considered logically and semantically coherent.*" for the criteria of Coherence. Additionally, those with non-linguistic backgrounds had certain terms clarified, such as the meaning of "prosaic" in the style-transfer task. Each evaluator rated 450 samples across all evaluations, consisting of 20 samples from 7 models (i.e., Flan-T5$_{2k}$, Flan-T5$_{13k}$, ByT5$_{2k}$, ByT5$_{13k}$, Baichuan, ChatGPT [i.e., GPT-3.5-Turbo], and the gold-standard) on 2 tasks (280 in total), 50 quality control examples, and 120 examples for our novel constrained decoding algorithm (20 of which used constrained base GPT-2, 20 of which came from our constrained fine-tuned GPT-2, and 20 of which came from unconstrained GPT-2 trained on 13k samples, with the equivalent set up for Baichuan).

6. Results (Topic-to-Twister and Style-Transfer)

6.1 Automatic Results

6.1.1 Topic-to-Twister. We present the results for automatic evaluation in the topic-to-twister task setting in Table 7 and Table 8. First, regarding the reference-based metrics (BLEU, ROUGE, and BERTScore), we see clear performance differences across our chosen models. On average across almost all metrics, we see the performance from worst to best ordered as GPT-2, DialoGPT, BART, ByT5, and Flan-T5 for our fine-tuned models. However, we see that Baichuan's performance is variable, mostly outperforming BART and underperforming Flan-T5 and ByT5. However, Baichuan performs

Table 7
Results of automatic evaluation on typical metrics for the topic-to-twister task setting. The mean length of the tongue twisters in the dataset is 35.65/34.14/14.98 words for train/evaluation/test, respectively. Results in **bold** represent the best performance for a given training data quantity, and underlined presents the second-best performance. Results followed by an asterisk * denote the overall best performance. The PPL of ByT5 is not applicable, as the tokens of ByT5 are characters whereas others are sub-words.

Model	PPL	B-1↑	B-2↑	B-3↑	B-4↑	Ro-1↑	Ro-2↑	Ro-L↑	Length
GPT-2$_{2k}$	25.04	0.0391	0.0165	0.0083	0.0043	5.1255	0.6704	4.8414	51.81
DialoGPT$_{2k}$	23.74	0.0453	0.0187	0.0091	0.0047	6.3366	1.0349	5.9843	57.72
BART$_{2k}$	3.98	0.0749	0.0256	0.0105	0.0040	8.1503	0.7608	7.4855	44.07
Flan-T5$_{2k}$	2.86	0.1055	0.0545	0.0325	0.0209	**11.2417**	**2.1528**	**10.3641**	25.60
ByT5$_{2k}$	–	**0.1440**	**0.0918**	**0.0639**	**0.0477**	6.2582	1.0329	5.9611	47.56
GPT-2$_{4k}$	24.84	0.0423	0.0168	0.0076	0.0037	5.7569	0.6818	5.5077	53.27
DialoGPT$_{4k}$	24.83	0.0422	0.0161	0.0071	0.0032	5.9712	0.6915	5.6415	55.91
BART$_{4k}$	4.09	0.0685	0.0215	0.0082	0.0031	7.6023	0.7364	6.9026	47.30
Flan-T5$_{4k}$	2.93	0.1090	0.0558	0.0328	0.0207	**10.5343**	**1.7701**	**9.8546**	24.25
ByT5$_{4k}$	–	**0.1397**	**0.0900**	**0.0634**	**0.0479**	6.8637	0.9744	6.4977	48.18
GPT-2$_{8k}$	28.58	0.0430	0.0143	0.0060	0.0026	5.6638	0.5385	5.2935	54.23
DialoGPT$_{8k}$	24.54	0.0458	0.0172	0.0076	0.0035	6.3555	0.7622	5.9901	56.59
BART$_{8k}$	4.33	0.0633	0.0195	0.0072	0.0029	6.8434	0.5874	6.1127	49.68
Flan-T5$_{8k}$	3.11	0.1194	0.0600	0.0346	0.0217	**10.7728**	**1.3758**	**10.0456**	22.97
ByT5$_{8k}$	–	**0.1531**	**0.0951**	**0.0647**	**0.0475**	6.2602	0.7142	6.0271	40.96
GPT-2$_{13k}$	28.77	0.0440	0.0152	0.0061	0.0025	5.9412	0.6024	5.5890	56.59
DialoGPT$_{13k}$	25.15	0.0504	0.0187	0.0078	0.0034	6.7274	0.8170	6.3148	60.05
BART$_{13k}$	4.16	0.0595	0.0152	0.0050	0.0017	6.6284	0.5090	5.9523	49.79
Flan-T5$_{15k}$	3.10	0.1189	0.0571	0.0319	0.0192	**10.0983**	1.1632	**9.2976**	23.87
ByT5$_{16k}$	–	**0.1609***	**0.0988**	**0.0660**	**0.0476**	6.4857	0.6073	6.2272	40.05
Baichuan+LoRA$_{13k}$	14.54	0.0463	0.0227	0.0131	0.0080	6.2215	**1.3878**	5.9212	51.76
ChatGPT	–	0.1577	0.1073*	0.0788*	0.0585*	26.2949*	13.2789*	23.7763*	24.46

worse than BART considering a range of metrics (B-1, Ro-1, Ro-L, BS-P, BS-R, BS-F1) when considering the same training data amount (13k). Flan-T5 and ByT5 alternate in performance, with the latter tokenizer-free model performing better on BLEU-based metrics, but worse on ROUGE. When specifically considering changes alongside an increase in training data, in Table 7 we see little to no improvement in reference-based metrics within the topic-to-twister setting, with performance decreasing as training data increases in some cases. However, it is pertinent to mention that reference-based metrics are imperfect for the task of creative language generation due to the one-to-many dilemma prevalent in many NLG tasks (Gupta et al. 2019), whereby there are numerous potential tongue twisters to generate from any given input topic. This is also particularly true when considering the TwistList 2.0 dataset, where the topics are often only related to the input phrase in a high-level conceptual fashion (as we do not enforce the generation of the topic phrase within the tongue twister itself). Finally, ChatGPT, when prompted in the same manner as our other models, demonstrates significantly higher performance than any of our fine-tuned models.

Regarding the readability and phonemic metrics presented in Table 8, we see a range of patterns. First, for IPO (formerly referred to as Init-PO) and PO, we see the GPT-2 based models "outperform" BART and the T5 models almost across the board. However, naive phoneme-overlap-based metrics do not take into account more sophisticated phonemic characteristics. When considering our new metrics based on

Table 8
Results of automatic evaluation on tongue twister related metrics for the topic-to-twister setting. Results in **bold** represent the best performance for a given training data quantity, and underlined presents the second-best performance. Results followed by an asterisk * denote the overall best performance. Brown Corpus Prose presents the scores of standard prose from the Brown Corpus from NLTK (across 602 sentences with a minimum length of 25 words).

Model	**BS-P↑**	**BS-R↑**	**BS-F1↑**	**IPO↓**	**PO↓**	**iPED↓**	**oPED↓**	**Re-D**	**Re-F**	**Re-G**	**Re-A**
GPT-2$_{2k}$	0.7543	0.8265	0.7883	**0.0849**	**0.0617***	3.8717	5.9131	17.08	45.79	48.71	59.52
DialoGPT$_{2k}$	0.7680	0.8310	0.7978	0.0954	0.0660	3.8308	5.9032	12.79	16.52	18.54	21.21
BART$_{2k}$	0.7938	0.8384	0.8153	0.3183	0.1848	4.5168	5.9219	14.21	11.70	14.12	11.51
Flan-T5$_{2k}$	**0.8075**	**0.8440**	**0.8249**	0.2129	0.1610	4.0828	5.9141	13.10	14.68	16.19	18.25
ByT5$_{2k}$	0.7705	0.8301	0.7982	0.1207	0.1021	**2.5398**	**5.9048**	14.61	28.54	30.20	38.35
GPT-2$_{4k}$	0.7648	0.8303	0.7958	**0.0881**	**0.0683**	4.1320	5.9440	13.51	23.08	25.14	28.89
DialoGPT$_{4k}$	0.7648	0.8300	0.7956	0.0984	0.0701	3.7237	5.9205	13.01	17.39	19.26	21.86
BART$_{4k}$	0.7962	0.8363	0.8156	0.2821	0.1620	4.7537	5.9693	12.77	12.46	14.11	13.69
Flan-T5$_{4k}$	**0.8071**	**0.8436**	**0.8244**	0.2089	0.1657	3.8280	**5.8899**	13.29	16.63	17.76	20.28
ByT5$_{4k}$	0.7636	0.8297	0.7941	0.1250	0.0987	**2.2248**	5.9197	16.0915	35.6761	36.9863	47.4017
GPT-2$_{8k}$	0.7712	0.8315	0.7998	0.1224	0.0841	4.3355	5.9751	12.57	14.93	16.30	18.00
DialoGPT$_{8k}$	0.7745	0.8327	0.8022	**0.1116**	**0.0775**	4.0111	5.8952	11.99	13.42	15.14	16.83
BART$_{8k}$	0.7928	0.8343	0.8129	0.2831	0.1675	4.9463	5.9693	11.97	12.36	13.71	13.86
Flan-T5$_{8k}$	**0.8160**	**0.8462**	**0.8304**	0.2378	0.1866	4.1542	**5.8669**	12.74	14.40	16.24	17.31
ByT5$_{8k}$	0.7711	0.8306	0.7986	0.1341	0.1087	**2.1024***	5.8804	16.37	31.98	33.88	42.47
GPT-2$_{13k}$	0.7748	0.8322	0.8021	0.1261	0.0859	4.2671	5.8985	12.06	13.46	15.21	16.64
DialoGPT$_{13k}$	0.7795	0.8336	0.8053	0.1249	0.0839	4.4665	5.9407	11.30	12.85	14.95	15.55
BART$_{13k}$	0.7925	0.8319	0.8115	0.2662	0.1489	4.8871	**5.8184***	12.14	11.75	13.57	13.83
Flan-T5$_{13k}$	**0.8181**	**0.8468**	**0.8319**	0.2568	0.1996	4.3122	5.8932	12.31	12.50	14.03	14.97
ByT5$_{13k}$	0.7742	0.8320	0.8009	0.1554	0.1217	**2.5520**	5.8374	15.77	31.22	34.00	39.33
Baichuan+LoRA$_{13k}$	0.7658	0.8259	0.7940	**0.0689***	**0.0657**	**2.6752**	5.9951	16.41	34.37	31.73	44.59
ChatGPT	0.8401*	0.8613*	0.8503*	0.3477	0.2991	4.0547	5.9153	9.83	9.37	10.53	11.48
Brown Corpus Prose	–	–	–	0.4870	0.2275	5.1431	5.9561	11.09	13.15	15.56	16.60

phonemic edit distance (iPED and oPED), we see less distinction across any of the presented models. However, one notable finding is that Baichuan performs significantly better than any of the other models trained on 13k samples in the word-initial IPO and iPED metrics, suggesting high levels of word-initial phonemic overlap across tokens when compared to other models. Likewise, we see that the tokenizer-free ByT5 model outperforms Flan-T5 by a significant margin regarding the phoneme-based metrics, suggesting that the finer-grained tokenization of ByT5 is preferable for identifying the sound patterns implicitly included in tongue twisters via grapheme combinations. ChatGPT, on the other hand, scores the highest on the IPO and iPED metrics (i.e., worst). This, however, is not in itself indicative of poor tongue twisters, but rather the model's focus on producing high-quality comprehensible and grammatical text, therefore being less likely to fall victim to degenerate patterns of repeating the same word over and over to achieve overlap. As with the reference-based metrics discussed earlier, raw reliance on phoneme-based metrics can also be misleading. For example, intuitively, prior works have demonstrated that high scores (i.e., lower values) can be achieved in word-initial-based metrics simply by repeating the same word, rather than producing a complex and entertaining tongue twister. Regarding readability scores, we see GPT-2 and Baichuan present high scores (i.e., high difficulty of readability) when compared to our other models. However, high readability metrics can be indicative of numerous things, including desirable behavior (using complex structures and sophisticated multi-syllabic vocabulary) as well as behavior that is not necessarily desirable (e.g., producing

Table 9
Results of automatic evaluation on typical reference-based metrics in NLG for the style-transfer setting. The mean length of the tongue twisters in the dataset is 35.65/34.14/14.98 words for train/evaluation/test, respectively. Results in **bold** represent the best performance for a given training data quantity, and underlined presents the second-best performance. Results followed by an asterisk * denote the overall best performance. The PPL of ByT5 is not applicable, as the tokens of ByT5 are characters whereas others are sub-words.

Model	**PPL**	**B-1↑**	**B-2↑**	**B-3↑**	**B-4↑**	**Ro-1↑**	**Ro-2↑**	**Ro-L↑**	**Length**
GPT-2$_{2k}$	15.59	0.1055	0.0738	0.0534	0.0397	15.2637	6.2852	14.8342	62.53
DialoGPT$_{2k}$	14.54	0.1001	0.0688	0.0491	0.0359	14.4003	5.7688	13.9574	67.65
BART$_{2k}$	1.96	0.2498	0.1797	0.1324	0.0997	25.4414	11.3534	24.8061	36.87
Flan-T5$_{2k}$	1.64	0.5597	0.4426	0.3591	0.2964	**48.7184**	**24.3008**	**47.9246**	15.18
ByT5$_{2k}$	–	**0.6770**	**0.5847**	**0.5210**	**0.4731**	46.8991	22.3860	46.0446	16.53
GPT-2$_{4k}$	15.15	0.1161	0.0837	0.0621	0.0472	16.7069	7.4608	16.3085	62.70
DialoGPT$_{4k}$	13.96	0.1099	0.0773	0.0565	0.0425	15.8087	6.7687	15.3450	65.72
BART$_{4k}$	2.00	0.2561	0.1859	0.1378	0.1041	26.3945	12.0011	25.8563	37.72
Flan-T5$_{4k}$	1.62	0.5500	0.4351	0.3532	0.2917	48.8662	24.6492	48.1269	14.99
ByT5$_{4k}$	–	**0.7177**	**0.6242**	**0.5596**	**0.5111**	**49.1321**	**24.7326**	**48.2396**	15.90
GPT-2$_{8k}$	14.26	0.1152	0.0833	0.0622	0.0476	16.7724	7.5582	16.3746	63.30
DialoGPT$_{8k}$	13.19	0.1096	0.0772	0.0572	0.0438	15.7687	7.0081	15.248	67.10
BART$_{8k}$	1.95	0.2448	0.1803	0.1358	0.1043	30.1371	14.4970	29.4553	37.50
Flan-T5$_{8k}$	1.63	0.5699	0.4571	0.3756	0.3135	**51.1949**	**26.9544**	**50.4617**	14.91
ByT5$_{8k}$	–	**0.7124**	**0.6224**	**0.5605**	**0.5136**	50.4737	25.9246	49.5851	15.85
GPT-2$_{13k}$	13.55	0.1229	0.0901	0.0683	0.0529	17.9146	8.4624	17.4801	60.71
DialoGPT$_{13k}$	12.84	0.1114	0.0796	0.0597	0.0462	16.2061	7.3163	15.7076	66.90
BART$_{13k}$	1.82	0.2298	0.1717	0.1309	0.1014	34.0662	16.9713	33.3075	37.78
Flan-T5$_{13k}$	1.61	0.5832	0.4704	0.3885	0.3258	52.7077	28.6604	51.8808	14.82
ByT5$_{13k}$	–	**0.7356***	**0.6465***	**0.5846***	**0.5376***	51.9886	27.7015	51.1895	15.55
Baichuan+LoRA$_{13k}$	5.15	0.6046	0.5006	0.4229	0.3618	**60.1989***	**37.1423***	**59.1040***	15.27
ChatGPT	–	0.3288	0.2167	0.1491	0.1057	39.3536	13.9654	34.9884	17.43

gratuitously long sentences). It is for this reason that we exclude an indicator of the preferred metric direction in the case of our readability metrics, yet include the scores for completeness.

6.1.2 Style-Transfer. We present the results for automatic evaluation in the style-transfer task setting in Table 9 and Table 10. Overall, we see much the same pattern as with the topic-to-twister setting, with performance ordering of DialoGPT, GPT-2, BART, ByT5, and Flan-T5 across our referenced metrics (again, with our T5-based models alternating in ranking). However, unlike in the previous setting, the style-transfer setting appears to favor Baichuan, which presents the highest scores on ROUGE-based referenced metrics (when considering models also trained on 13k samples), whereas ByT5 performs the best on BLEU. Additionally, scores across the board are higher than seen in the topic-to-twister setting, but this is to be expected as the style-transfer task setting provides a structure for the generated tongue twister based on the length and word choices present in the original. Consequently, increased amounts of overlap between the desired output and the gold reference are expected due to not all words requiring modification. In contrast to the topic-to-twister setting, however, we do not see ChatGPT outperform all models, rather, it is beaten by Baichuan and Flan-T5 in most cases. In terms of the

Table 10
Results of automatic evaluation on tongue-twister-related metrics for the style-transfer setting. Results in **bold** represent the best performance for a given training data quantity, and underlined presents the second-best performance. Results followed by an asterisk * denote the overall best performance. Brown Corpus Prose presents the scores of standard prose from the Brown Corpus from NLTK (across 602 sentences with a minimum length of 25 words).

Model	BS-P↑	BS-R↑	BS-F1↑	IPO↓	PO↓	iPED↓	oPED↓	Re-D	Re-F	Re-G	Re-A
GPT-2$_{2k}$	0.8170	0.8662	0.8405	0.1264	0.0830	4.7213	5.9688	10.66	7.45	9.52	6.94
DialoGPT$_{2k}$	0.8172	0.8634	0.8394	**0.1177***	**0.0786***	4.8873	5.9570	9.92	7.66	9.76	7.74
BART$_{2k}$	0.8162	0.9031	0.8570	0.5112	0.2474	4.5594	**5.8120**	14.76	12.21	14.13	11.47
Flan-T5$_{2k}$	**0.9281**	**0.9275**	**0.9277**	0.5465	0.4290	4.7362	5.9696	10.51	6.08	7.77	5.07
ByT5$_{2k}$	0.9166	0.9171	0.9166	0.4392	0.3949	**4.2451**	5.9703	10.78	6.81	8.72	5.90
GPT-2$_{4k}$	0.8247	0.8726	0.8477	**0.1221**	**0.0834**	4.7674	5.9775	10.88	7.02	8.93	6.15
DialoGPT$_{4k}$	0.8213	0.8668	0.8432	0.1280	0.0847	4.8948	5.9825	10.34	7.83	10.16	7.35
BART$_{4k}$	0.8157	0.9026	0.8565	0.4028	0.2267	**3.8435**	**5.7562***	16.74	14.10	15.72	14.19
Flan-T5$_{4k}$	**0.9262**	**0.9266**	**0.9263**	0.5362	0.4290	4.6793	5.9743	10.72	6.10	7.83	5.05
ByT5$_{4k}$	0.9198	0.9199	0.9197	0.4436	0.4011	4.2007	5.9553	10.47	6.40	8.03	5.59
GPT-2$_{8k}$	0.8231	0.8714	0.8463	**0.1188**	**0.0821**	4.7338	5.9779	10.83	7.30	9.23	6.58
DialoGPT$_{8k}$	0.8183	0.8638	0.8402	0.1329	0.0876	4.9483	5.9787	10.10	8.23	10.47	8.03
BART$_{8k}$	0.8073	0.9073	0.8538	0.4451	0.2620	5.1885	**5.8138**	14.46	13.17	13.46	11.00
Flan-T5$_{8k}$	**0.9289**	**0.9290**	**0.9288**	0.5262	0.4289	4.6310	5.9707	10.67	5.95	7.67	4.99
ByT5$_{8k}$	0.9208	0.9219	0.9212	0.4549	0.4063	**4.2830**	5.9598	10.68	6.52	8.04	5.85
GPT-2$_{13k}$	0.8258	0.8738	0.8489	**0.1256**	**0.0870**	4.7119	**5.9536**	11.65	8.59	10.84	7.73
DialoGPT$_{13k}$	0.8216	0.8676	0.8437	0.1348	0.0891	5.0811	5.9733	10.51	9.01	11.03	8.79
BART$_{13k}$	0.7903	0.9141	0.8470	0.4201	0.3216	**4.0013***	6.1303	17.97	13.25	16.91	9.82
Flan-T5$_{13k}$	0.9311	0.9309	0.9309	0.5226	0.4288	4.5770	5.9681	10.64	5.98	7.63	5.06
ByT5$_{13k}$	0.9236	0.9246	0.9240	0.4619	0.4128	4.2982	5.9634	10.73	6.35	7.93	5.67
Baichuan+LoRA$_{13k}$	**0.9442***	**0.9411***	**0.9425***	0.4908	0.4236	4.4789	5.9771	9.98	5.60	7.15	4.62
ChatGPT	0.8851	0.8898	0.8873	0.5495	0.3948	4.7725	5.9445	10.37	7.67	9.06	8.25
Brown Corpus Prose	–	–	–	0.4870	0.2275	5.1431	5.9561	11.09	13.15	15.56	16.60

performance difference on these metrics as the amount of available training data is increased, unlike the topic-to-twister setting we see a more clear growth in performance on reference-based metrics alongside training data in the majority of cases. Regarding the readability and phonemic metrics presented in Table 10, we again see similar patterns with GPT-2 based models scoring well on the naive phoneme-based metrics (IPO/PO), but all models performing similarly when regarding the more informed iPED/oPED measures. Regarding readability, scores overall are seen to be lower than in the topic-to-twister setting, suggesting more legible text. On one hand, this may indicate that the style has failed to transfer, with the paraphrase representing a well-written standard non-literary text (and therefore the model has effectively resorted to auto-encoding). On the other hand, this may also be an artifact of following the original structure of the non-literary paraphrase, therefore avoiding unnaturally long sentences and nonsensical outputs.

6.2 Human Evaluation

The results of human evaluation for the topic-to-twister setting are presented in Table 11, and the results for the style-transfer setting are in Table 12.

6.2.1 Topic-to-Twister. First, for the topic-to-twister setting in Table 11, we can see that the highest scores for all criteria go to the human-authored "golden" samples, or those

Table 11
Results of human evaluation in the topic-to-twister task setting. The best scores are in **bold**, and the second-best are underlined. We calculate Fleiss' Kappa for each metric, and we mark the extent of agreement with the following markings: * fair agreement; ** moderate agreement; *** substantial or almost perfect agreement.

Score (1 to 5)	Trained Topic-to-Twister						
	Flan-T5$_{2k}$	Flan-T5$_{13k}$	ByT5$_{2k}$	ByT5$_{13k}$	Baichuan$_{13k}$	ChatGPT	Golden
Relevance	2.077***	1.625***	2.180**	2.070**	1.688**	**4.824****	<u>4.647</u>**
Articulation	1.800**	1.882**	3.050**	3.290***	1.176**	<u>2.667</u>**	**3.375***
Fluency	2.000**	3.462**	2.290**	3.380**	1.450**	<u>4.632</u>**	**4.944****
Coherence	1.200**	2.133**	1.610**	1.930**	1.118**	<u>4.333</u>**	**4.444*****
Entertainment	1.200*	1.833*	1.620**	2.070**	1.000*	**3.267***	<u>3.077</u>**
Overall	1.063*	1.888*	1.800**	2.300**	1.316**	<u>3.538</u>**	**3.909****

Table 12
Results of human evaluation in the style-transfer task setting. The best scores are in **bold**, and the second-best are underlined. We calculate Fleiss' Kappa for each metric, and we mark the extent of agreement with the following markings: * fair agreement; ** moderate agreement; *** substantial or almost perfect agreement.

Score (1 to 5)	Style-Transfer						
	Flan-T5$_{2k}$	Flan-T5$_{13k}$	ByT5$_{2k}$	ByT5$_{13k}$	Baichuan$_{13k}$	ChatGPT	Golden
Relevance	4.467*	4.714**	4.090*	4.160*	<u>4.882</u>**	4.692**	**5.000***
Articulation	1.462**	2.231**	3.710**	3.520*	2.250**	<u>2.471</u>**	**3.313****
Fluency	4.611**	4.895***	4.150**	4.270**	4.800*	**5.000****	<u>4.950</u>**
Coherence	4.188**	<u>4.733</u>**	3.500**	3.550**	4.375**	3.929**	**4.786***
Entertainment	1.733*	2.000*	3.400**	3.240*	<u>2.846</u>*	2.583*	**3.455****
Overall	2.308*	3.000*	3.500**	3.510*	3.333*	<u>3.500</u>*	**3.941****

generated with ChatGPT. With a rating of "3" being considered the midpoint for "neither agree nor disagree" with the given criteria statements, it is evident that our fine-tuned unconstrained generation models struggle with the open-ended topic-to-twister task setting. However, when investigating the fine-tuned model performance we do see some patterns start to emerge that indicate the benefit of having such an extensive dataset as TwistList 2.0. For instance, Flan-T5 benefits from additional training samples, particularly regarding the metrics of Fluency and Coherence, and moderately in Entertainment and the holistic Overall rating. These findings for Fluency and Coherence are intuitive, as additional training samples increase the likelihood of generating grammatical and semantically coherent outputs due to the increased training data through which to learn these patterns. On the other hand, Articulation and Entertainment refer to more creative-language-specific metrics that are more abstract, and consequently difficult to learn from the training data. Relevance is the only metric shown to decrease when moving from 2k training samples to 13k, and we hypothesize that this is due to the increased wealth of training data that contains a more abstract link between the input topic and the generation, therefore decreasing the likelihood of the input words being directly present in the output (which is a straightforward way of performing well on Relevance metrics). Overall, however, we see ByT5 is preferable to Flan-T5, outperforming it in human evaluation on the criteria of Relevance, Articulation, Entertainment, and Overall, but underperforming in regard to Fluency and Coherence in the 13k instance. This additionally sheds some light on how humans perceive quality tongue twisters, with articulation difficulty being a more integral feature for a tongue twister than grammatical validity and semantic coherence. Finally, we see Baichuan struggle immensely with the topic-to-twister task setting (which is explored further in §9), frequently opting to repeat the input topic phrase continuously, therefore artificially increasing Relevance scores, but performing poorly on all other metrics.

6.2.2 Style-Transfer. On the other hand, we see better performance across the board for the style-transfer setting as facilitated by the additional high-quality paraphrases we include in TwistList 2.0. We hypothesize that the reason for this is that style-transfer requires already having access to a well-formed input, and additionally acts as an extended, very prescriptive form of an input topic, where the entire tongue twister is predefined in structure and semantics. As a result, we see Baichuan outperform ChatGPT on the criteria of Entertainment and Relevance, whereas Flan-T5 trained on the 13k split outperforms ChatGPT regarding semantic coherence of the output. Interestingly, we observe that ByT5 trained on only 2k examples performs more closely to the 13k version than is seen in Flan-T5, suggesting the alternative tokenization approach allows ByT5 to learn the relevant patterns from fewer examples. Importantly, we see Baichuan perform much better in the style-transfer task than in the topic-to-twister setting, suggesting that Baichuan requires much more explicit instruction to generate high-quality outputs and understand a given prompt.

6.3 Human vs. Automatic Metrics

6.3.1 Human-Machine Correlation. In order to see the effectiveness of our selected automatic metrics, we calculated the Spearman correlations between our 8 reference-free metrics, including readability (i.e., Re-D, Re-F, Re-G, and Re-A) and our phonetic metrics (i.e., IPO, PO, iPED, and oPED), against the human evaluation ratings for all criteria (i.e., relevance, articulation, fluency, coherence, entertainment, and overall). Correlations are presented in Table 13. To evaluate the predictive power of the automatic

Table 13
Spearman correlation coefficients for each human evaluation criterion with reference-free automatic metrics. We take the mean score across evaluators for the combined unconstrained topic-to-twister and style-transfer task settings.

	Automatic Metrics							
	Re-D	**Re-F**	**Re-G**	**Re-A**	**IPO**	**PO**	**iPED**	**oPED**
Relevance	−.395	−.442	−.423	−.494	.394	.496	−.004	.104
Articulation	−.218	−.271	−.224	−.218	−.063	−.087	−.226	.110
Fluency	−.400	−.590	−.564	−.630	.516	.674	−.001	−.013
Coherence	−.320	−.482	−.551	.482	−.486	.622	−.003	.064
Entertainment	.301	−.287	−.267	−.278	.151	.262	−.077	−.106
Overall	.342	.386	.372	.386	.211	.361	−.129	.055

metrics for human ratings, we developed a standard multiple linear regression model for each criterion. The model is defined as follows:

$$y_i = \beta_0 + \sum_{j=1}^{p} \beta_j X_{ij} + \epsilon_i \tag{1}$$

where y_i represents the human score, β_0 is the intercept, β_j are the coefficients for the p automatic metrics, and ϵ_i is the error term. No regularization was applied to the model. The R^2 values, which indicate the proportion of variance explained by the automatic metrics, were validated through 5-fold cross-validation to ensure the stability and significance of the predictions. The average R^2 values across the folds were as follows: Fluency ($R^2 = .553$), Coherence ($R^2 = .496$), Relevance ($R^2 = .393$), Overall ($R^2 = .260$), Articulation ($R^2 = .241$), and Entertainment ($R^2 = .152$). Furthermore, all coefficients in the models were found to be statistically significant, with p-values below 0.01 ($\alpha = 0.01$), indicating that each of the automatic metrics significantly contributes to the prediction of human ratings.

Overall, we intuitively see the Entertainment criterion being the hardest to predict due to the inherent subjectivity of this criterion. On the other hand, we see our naive phonemic metrics (IPO and PO) demonstrate moderate correlations with Relevance, Fluency, and Coherence. This is due to the high relevance of tongue twisters often being seen in examples where the input topic is simply repeated, which results in high levels of phoneme overlap. Similarly, high fluency scores are given to more sentences that better reflect standard non-literary text, which consequently score lower on IPO and PO, and less coherent outputs are often produced by repeating the same word. Moreover, our "informed" phonemic metrics (iPED/oPED) show little correlation with human results on these criteria, but iPED demonstrates evidence of a correlation with articulatory difficulty. However, the articulatory difficulty still remains challenging to predict, even from these phonemic metrics. We hypothesize that this may be related to the "visual tongue twister" effect (McCutchen and Perfetti 1982). This is due to human evaluation being performed online and asynchronously, where we cannot force participants to speak aloud each tongue twister. Consequently, we hypothesize that the naive metrics may correlate better with human judgments as human ratings were confounded by the visual impact of the tongue twister (for example, seeing a particular grapheme repeated numerous times representing the same sound). Furthermore, there are additional effects from the influence of other factors such as fluency and coherence affecting articulation due to violating expectations and reducing legibility. Consequently, iPED/oPED should be used as indicators of text resembling a tongue twister, but not as a holistic metric for overall quality. It is clear from comparison with standard non-literary text that the phonetic metrics are able to differentiate the specific characteristics of tongue twisters from that of standard text, but phonemic complexity is not the sole contributor to the perception of articulatory difficulty.

6.3.2 GPT-4o "Human" Evaluation. We additionally perform evaluation on the same samples presented to human evaluators in the unconstrained topic-to-twister and style-transfer settings using GPT-4o via prompting the model with the same rubric presented to human evaluators (see Appendix A).[12] Overall, we see moderate-to-high correlation

12 Specifically, *gpt-4o-2024-05-13* via the API.

between model scores and human-assigned scores for all criteria: Relevance ($\rho = .671$), Articulation ($\rho = .653$), Fluency ($\rho = .768$), Coherence ($\rho = .658$), Entertainment ($\rho = .716$), and Overall ($\rho = .776$).[13] This demonstrates that our human evaluation rubric is clear and well defined, and can be effectively followed by state-of-the-art LLMs.

7. Tongue Twister Generation with Phoneme Aware Constrained Decoding

In the following section, we present work on a constrained decoding-based approach to tongue twister generation. In contrast to §5, here we focus exclusively on the topic-to-twister task setting due to the text-continuation nature of our decoding approach. The benefit of this algorithm, in contrast to the fine-tuned models presented in §5, is that constrained decoding *guarantees* that only desirable tokens appear in the output due to the layering of hard phoneme-based constraints as previously discussed. Additionally, due to how this system interacts with language model token predictions, this process can be applied to any autoregressive language model, including both pre-trained base models and further fine-tuned models.

7.1 Task Definition

For a given input prompt we aim to generate a tongue twister T, whereby T constitutes a sequence of words $\{w_1, w_2, \dots w_n\}$. In contrast with the previous section, in this task setting T is a continuation of the input prompt that we generate token by token, evaluating the language model's next token predictions at each step. As per §5, the generated output must satisfy the following constraints: (1) the output should be semantically related to the input topic phrase; (2) the output should show maximal levels of phonemic overlap across tokens; and (3) the output should be grammatically valid.

7.2 Phoneme-Aware Constrained Decoding Module (PACD)

We present an outline of our Phoneme-Aware Constrained Decoding algorithm (PACD) in Algorithm 1.[14] To summarize, for every starting prompt s in our test set S, we firstly perform grapheme-to-phoneme conversion $G2P$ with the g2p-en package and extract the initial phoneme of the first word in the topic phrase part of s in order to increase the likelihood of retrieving a semantically related output (with the phoneme denoted as ph_1). However, where the selected phoneme is not a valid consonant, we randomly select a phoneme from a list of phonotactically legal word-initial consonant phonemes for English, *WIP*. Following this, we calculate the weighted PED between ph_1 and all other legal word-initial phonemes and select the lowest scoring (i.e., most similar) as our secondary phoneme ph_2 (analogous to the system in §4 for TwisterLister). Following this, we autoregressively generate new tokens up to the limit defined by *max_length* based on numerous criteria. To do this, we feed the starting prompt s into our language model of choice, *LM*, and retrieve the next token probabilities P. Then, in descending order (i.e., most-probable to least-probable next token) we iterate through predictions $p \in P$ until specific criteria are met. First, to increase the likelihood of generating

13 All significant at $\alpha = .01$.

14 PACD is intended to be pronounced as "packed".

Algorithm 1 Phoneme-Aware Constrained Decoding (PACD)

```
1: for each $s$ in $S$ do
2:     $ph_1$ = G2P(topic in $s$)[0]
3:     $ph_2 = \arg\min(PED(ph_1, WIP \setminus \{ph_1\}))$
4:     while len(s*) < max_length do
5:         Retrieve next word probabilities $P = \{p_1, \ldots, p_n\}$ from $LM(s^*)$
6:         for each $rank, p$ in ENUMERATE($P$) do
7:             if $p$ in $F$ and $rank \leq$ function_window then
8:                 append $p$ to $s^*$
9:                 break
10:            end if
11:            $candidates$ = []
12:            if len($p$) > $min\_stem\_length$ and G2P($p$)[0] == $ph_1$ or $ph_2$ then
13:                append $p$ to $candidates$
14:                $temp\_prompt = s^* + p$
15:                for i in range(4) do
16:                    $next\_token$ = $LM(temp\_prompt)$
17:                    if $next\_token$.isalpha() and $next\_token$[0] != “ ” then
18:                        $longest$ = “ ”.join($candidates$)
19:                        $longest$ += $next\_token$
20:                        append $longest$ to $candidates$
21:                    end if
22:                    if $next\_token[-1]$ == “ ” then
23:                        break
24:                    end if
25:                end for
26:                $temp\_prompt$ = “ ”
27:                for $candidate$ in $candidates$.sort(longest-to-shortest) do
28:                    if $candidate \in D$ and COUNT($candidate$ in $s^*$) < $max\_repetition$ then
29:                        append $candidate$ to $s^*$
30:                    end if
31:                end for
32:            end if
33:        end for
34:    end while
35: end for
```

grammatical output, if a function word (such as an article, pronoun, conjunction, preposition, or auxiliary verb) from our function word list F is within the range defined by *function_window*, we allow it to generate. For example, when generating the first word, if *function_window* is set to 3, and “The” is the token with the 2nd highest probability, we allow it to generate as it is within our allowed range of top-3.

To account for subword tokenization in non-function words, we next check that the predicted token is longer than the limit defined by *min_stem_length*. We do this as we find the result of not limiting this to be a reliance on outputting the grapheme that most closely corresponds to a desired phoneme (rather than a sequence that better resembles a morpheme), significantly increasing inference time and decreasing output quality. We then use our phoneme constraints by feeding the predicted word stem p

into a grapheme-to-phoneme model *G2P* and comparing the first phoneme to ph_1 and ph_2, continuing if it matches either. Consequently, in this stage, we have ensured that generated words are either closed-class grammatical function words or start with one of the two phonologically similar sounds selected in lines 2–3 of Algorithm 1. Following this, we optionally engage the subword loop (lines 15–25 in Algorithm 1). Within this loop, we temporarily append our candidate word stem to the current prompt s^* to create *temp_prompt*. Following this, we feed *temp_prompt* to the language model *LM* and take only the token with the highest probability, *next_token*. We then check that *next_token* is alphabetical and does not start with whitespace (as this would indicate the model was predicting a new word, rather than a continuation). If this is the case, we append it to *temp_prompt* and perform the loop again (up to 4 times, allowing words that consist of 1–5 subwords). Within this loop, we build a list of potential words, *candidates*, by appending the concatenated subwords (e.g., ["anti", "antidis", "antidisestablish", "antidisestablishment", "antidisestablishmentarian"]) We terminate this loop early if a predicted *next_token* ends with whitespace, as this indicates that the model has predicted the end of the current word. Once we have our list of *candidates*, we iterate through them from the longest (i.e., consisting of the most subwords) to the shortest, assessing the following criteria.

First, we check that each $candidate \in candidates$ is longer (in characters) or equal to the length defined by *min_word_length* and that $candidate \in D$, where D is the English dictionary as defined by the Enchant Python package.[15] Once this check is complete, we ensure that we have not already generated this specific *candidate* more times than permitted by *max_repetition*, in order to avoid falling into the perpetual loop of repeating the same words that language models are prone to (see §9). Here we use s^* to denote the starting prompt s with *newly* generated tokens appended, which is to say that $s^* - s$ equals only the LM-generated words. Finally, if no *candidate* meets the criteria, we increase *rank* by looking at the next-best prediction in P until the vocabulary is exhausted. In the case where no suitable candidates exist in the vocabulary, we simply move on to the next s.

To illustrate the algorithm with an example, consider S to consist of two input topics s: [fun, sadness]. For the first example ("fun"), we select ph_1 by performing G2P on the topic, returning /fʊn/, and select the word-initial /f/ as ph_1. We then decide ph_2 by selecting the next phoneme that has the lowest phonemic edit distance to /f/, returning /v/ as ph_2. Following this, we feed the full prompt *"Generate a tongue twister on the topic of 'fun'."* to the language model, and retrieve the next-token probabilities P. For example, P could be $\{1 : The, 2 : It, 3 : A...\}$, where the set is the length of the decoded vocabulary. We then iterate through the predictions until a word meets our criteria. For instance, the most likely continuation, "The", is in the function word list F and within the *function_window* due to being at *rank* 1, so we append it to the prompt and now have *"Generate a tongue twister on the topic of 'Fun'. <u>The</u>"*, which we now denote $s*$. We then feed this extended prompt into the language model to retrieve the second word, where P may look like *{1: grey, 2: big, 3: fun}*. Here, options in *rank* 1 and 2 ("grey" and "big") do not start with ph_1 or ph_2 and are also not function words in F. However, the word in rank 3, "fun" is transcribed phonemically as /fʊn/, where the initial phoneme /f/ matches ph_1, so we enter the subword loop and find the candidate "funniest", allow it to generate, and append it to the prompt, resulting in *"Generate a tongue twister on the topic of 'Fun'. <u>The funniest</u>"*. We repeat this until we generate new tokens up to *max_length*,

15 Available at `https://pypi.org/project/pyenchant/`.

and then start the process again for the remaining topic in *S*, which is "sadness". We additionally make sure that we do not generate the same word more than once, as determined by *max_repetition*, and we do not generate words shorter in length than *min_word_length*.

7.3 Constrained Models

To demonstrate the effectiveness of our decoding module, we utilize 2 decoder-only autoregressive language models as our *LM*: **GPT-2** (Radford et al. 2019) and **Baichuan** (Yang et al. 2023), to which our module will be applied on top. In addition, we investigate to what extent fine-tuning a model towards tongue twister generation is beneficial, by additionally using our fine-tuned GPT-2 and Baichuan from §5, referred to herein as **GPT-2$_{13k}$** and **Baichuan$_{13k}$**. Importantly, we assess only these models fine-tuned on the largest amount of data, 13k.

Regarding the other settings for PACD, we set *max_length* to 30 (as a sensible midpoint generation length ascertained from Table 7), *function_window* to 1, and implement *F* as the NLTK stopwords list with all punctuation removed. Additionally, we set *min_stem_length* to 2 and *min_word_length* to 3 (as all standard 1- or 2-letter words $\{I, a, I'm, am, at, in, up, on\} \in F$). Additionally, *max_repetition* is set to 1, in effect banning wholesale repetition (though allowing plural/singular forms, and case variants) to avoid the patterns seen in standard autoregressive models. Finally, we use the *g2p-en* package for our *G2P* model, as in the creation of TwisterLister (§4.4). Finally, we only decode the top 2,500 predictions in each timestep rather than the entire vocabulary in order to speed up inference significantly, as it is rare to select tokens ranked below this point. Importantly, due to the computational cost of our algorithm, we load Baichuan using 8-bit quantization to make inference possible. Overall, for GPT-2, PACD takes approximately 5–10 seconds to generate a 30-word tongue twister (with or without subword generation), whereas Baichuan takes 10–15 seconds when generating full words only, and 30–100+ seconds when allowing subwords on a consumer CPU (i5 9600k). We posit future work on the parallelization of elements of PACD to be more computationally-efficient and take advantage of the GPU.

8. Results (PACD)

We perform automatic evaluation in the same manner as §6.1, and report both reference-based (BLEU/ROUGE/BERTScore) and unreferenced metrics (Init-PO/PO/iPED/oPED and the readability metric suite). We additionally perform human evaluation in the same manner as §6.2 and with the same evaluators, following the protocol for the topic-to-twister task setting. Each evaluator is presented with 20 examples (using the same inputs as in §6.2) from base GPT-2 and Baichuan with the addition of PACD, or fine-tuned GPT-2$_{13k}$ and Baichuan$_{13k}$ with and without PACD.

8.1 Automatic Evaluation

The results of the automatic evaluation for the constrained decoding approach (PACD) are presented in Table 14 for referenced metrics and Table 15 for unreferenced metrics. First, for GPT-2, regarding the reference-based metrics, surprisingly we see fine-tuned GPT-2 with the addition of our constrained decoding module (*GPT-2$_{13k}$ -w*) outperform the standard fine-tuned model (*GPT-2$_{13k}$ -w/o*) on B-1 and B-2, in addition to all ROUGE-based metrics (Ro-1, Ro-2, and Ro-L). The exception here is for the higher-order BLEU

Table 14
Results of automatic evaluation on typical reference-based metrics in NLG for the constrained decoding approach to tongue twister generation in the topic-to-twister task setting. Results in **bold** represent the best performance, and underlined results show the second-best performance. -w/o denotes without constraints, -w denotes the addition of our PACD module, and -ws denotes the addition of PACD with subword generation enabled.

Model	B-1↑	B-2↑	B-3↑	B-4↑	Ro-1↑	Ro-2↑	Ro-L↑
GPT-2$_{13k}$ -w/o	0.0440	0.0152	0.0061	0.0025	5.9412	0.6024	5.5890
GPT-2$_{13k}$ -w	**0.0832**	0.0170	0.0046	0.0013	**8.6673**	0.9067	**7.5011**
GPT-2$_{13k}$ -ws	0.0584	0.0045	0.0003	0.0000	0.0089	0.0000	0.0792
GPT-2 -w	0.0759	0.0142	0.0032	0.0010	8.0608	0.6403	6.8765
GPT-2 -ws	0.0594	0.0059	0.0009	0.0001	0.0940	0.0121	0.0790
Baichuan$_{13k}$ -w/o	0.0463	**0.0227**	**0.0131**	**0.0080**	6.2215	**1.3878**	5.9212
Baichuan$_{13k}$-w	0.0493	0.0041	0.0004	0.0000	0.0846	0.0081	0.0677
Baichuan$_{13k}$ -ws	0.0525	0.0051	0.0006	0.0001	0.0940	0.0106	0.0742
Baichuan -w	0.0547	0.0054	0.0009	0.0002	0.0885	0.0109	0.0750
Baichuan -ws	0.0600	0.0080	0.0019	0.0005	0.1010	0.0175	0.0846

Table 15
Results of automatic evaluation on tongue-twister-related metrics for the constrained decoding approach to tongue twister generation in the topic-to-twister task setting. Results in **bold** represent the best performance and underlined presents the second-best performance. -w/o denotes without constraints, -w denotes with the addition of our PACD module, and -ws denotes the addition of PACD with subword generation enabled. Brown Corpus Prose presents the scores of standard prose from the Brown Corpus from NLTK (across 602 sentences with a minimum length of 25 words).

Model	BS-P↑	BS-R↑	BS-F1↑	IPO↓	PO↓	iPED↓	oPED↓	Re-D	Re-F	Re-G	Re-A
GPT-2$_{13k}$ -w/o	0.7748	0.8322	0.8021	**0.1261**	**0.0859**	4.2671	5.8985	12.06	13.46	15.21	16.64
GPT-2$_{13k}$ -w	0.8056	0.8207	0.8129	0.2305	0.2317	3.0376	5.8930	13.88	14.58	17.58	18.77
GPT-2$_{13k}$ -ws	0.8018	0.8267	0.8139	0.1724	0.2287	1.7252	5.7686	11.60	12.24	15.59	17.20
GPT-2 -w	0.7986	0.8208	0.8094	0.1689	0.2110	1.6529	5.7699	11.36	12.66	15.76	15.91
GPT-2 -ws	0.8119	0.8258	0.8186	0.2469	0.2614	3.1642	5.8991	9.02	10.45	14.16	13.65
Baichuan$_{13k}$ -w/o	**0.9442**	**0.9411**	**0.9425**	0.4908	0.4236	4.4789	5.9771	9.98	5.60	7.15	4.62
Baichuan$_{13k}$ -w	0.7915	0.8202	0.8053	0.1629	0.2416	1.4324	5.6876	10.72	11.18	13.52	15.18
Baichuan$_{13k}$ -ws	0.7887	0.8215	0.0742	0.1512	0.2194	**1.2898**	**5.6708**	11.95	12.59	14.82	17.21
Baichuan -w	0.8014	0.8204	0.8106	0.2102	0.2461	2.4449	5.8094	9.08	10.55	14.26	14.25
Baichuan -ws	0.7984	0.8217	0.8096	0.1992	0.2339	2.3284	5.8016	9.89	11.37	14.86	15.38
Brown Corpus Prose	—	—	—	0.4870	0.2275	5.1431	5.9561	11.09	13.15	15.56	16.60

measures, B-3 and B-4, where the unconstrained model achieves higher overlap. Base GPT-2 also benefits from the addition of our PACD module (*GPT-2 -w*), outperforming the unconstrained fine-tuned model on the recall-based ROUGE metrics. Observing the unreferenced results in Table 15, performance ordering varies for the BERTScore semantic metrics (BS-P, BS-R, and BS-F1), with the unconstrained fine-tuned model (*GPT-2$_{13k}$ -w/o*) trading places with the constrained equivalents (*GPT-2$_{13k}$ -w and* -ws) for the most performant. However, base GPT-2 with the addition of PACD (*GPT-2 -w*) consistently comes in second place for these metrics. Furthermore, we see the original fine-tuned unconstrained model perform the best when considering the original naive phoneme-based metrics IPO and PO, whereas the un-fine-tuned, yet constrained model, *GPT-2*

-w, places second. It is here that the less naive newly presented phoneme-based metrics, iPED and oPED, demonstrate their usefulness. For instance, although fine-tuned GPT-2 without any constraints (GPT-2_{13k} -w/o) performed best on the naive metrics (IPO/PO), it performed the worst on the more linguistically informed metrics (iPED/oPED). This is because good tongue twisters exploit the relationships between similar sounds, whereas IPO/PO penalize these transitions as being low-quality; our informed metrics reflect favorably on such transitions, penalizing them less than transitions between weakly related phonemes.

When looking at Baichuan, we see that the unconstrained fine-tuned model is the highest overall scorer on B-2, B-3, B-4, and Ro-2, as well as all BERTScore measures (BS-P, BS-R, and BS-F1). Interestingly, when analyzing the impact of the addition of PACD, we see the non-fine-tuned Baichuan with subword generation enabled (*Baichuan -ws*) to outperform all other versions of Baichuan that contain PACD, even without being fine-tuned on the specific style of text we are aiming to generate (however, scores are low overall). Regarding the phonemic metrics, we again see the benefit of PACD in reducing the scores on all phoneme-based metrics by successfully increasing sound overlap. Additionally, we see fine-tuned Baichuan with the addition of PACD (*Baichuan*$_{13k}$ *-w* and *-ws*) to outperform the non-fine-tuned models in the phonetic metrics. Finally, the addition of PACD to Baichuan can be seen to lead to a significant increase in readability scores (i.e., an increase in reading difficulty). However, these scores still remain below the formal non-literary text of the Brown Corpus.

These results demonstrate the effectiveness of our constrained decoding approach that takes into consideration which word-initial phonemes would be the best to center generation around to encourage mispronunciation. Additionally, overall we see similar performance between our models using full-word and subword versions of PACD.

8.2 Human Evaluation

Human evaluation is performed identically to the evaluation reported in §6.2 (and with the same evaluators). Importantly, however, due to enforcing a 30-word output length, we ask evaluators to not penalize generations on the criteria of "Fluency" for being cut off prematurely. Additionally, due to the similarity in the outputs with and without sub-word generation for PACD as seen in §9, we only perform human evaluation on the outputs of PACD with subword generation disabled, to minimize evaluator fatigue and potential acquiescence bias due to seeing similar results for either method. The results of human evaluation on outputs using our constrained generation PACD module are presented in Table 16. We additionally perform human evaluation on GPT-2 and Baichuan trained on 13k samples (the former of which was excluded from §6.2) in order to have a point of comparison without the addition of PACD (referred to as *GPT-2*$_{13k}$ *-w/o* and *Baichuan*$_{13k}$ *-w/o*, respectively).

Regarding GPT-2, we see a clear benefit from the addition of our constrained decoding module. Firstly, across all evaluation criteria, either base (i.e., vanilla) or fine-tuned GPT-2 receives the highest scores (indicated in bold) when the PACD module is applied. However, we do see fine-tuned GPT-2 without the addition of PACD (*GPT-2*$_{13k}$ *-w/o*) outperforming the equivalent model with PACD enabled on the criteria of Fluency and Coherence, which evaluate grammar and semantics, respectively. One explanation for this is that GPT-2 (overall) has been shown to perform poorly in the topic-to-twister setting when considering coherent and grammatical outputs (see also §9). Consequently, the addition of even more restrictive decoding rules brought about by PACD is slightly detrimental to the general quality of the output. On the other hand, non-fine-tuned

Table 16
Results of human evaluation in the topic-to-twister task setting using constrained decoding. -w denotes to models using our decoding algorithm, and -w/o denotes fine-tuned models without this additional module. The best scores are in **bold**, and the second-best are underlined. We calculate Fleiss' Kappa for each metric, and we mark the extent of agreement with the following markings: * fair agreement; ** moderate agreement; *** substantial or almost perfect agreement.

Score (1 to 5)	Constrained Topic-to-Twister					
	GPT-2$_{13k}$ -w/o	GPT-2$_{13k}$ -w	GPT-2 -w	Baichuan$_{13k}$ -w/o	Baichuan$_{13k}$ -w	Baichuan -w
Relevance	2.16***	<u>2.44</u>***	2.42***	**3.04****	2.39**	2.41**
Articulation	2.80***	**3.85*****	<u>3.69</u>***	3.21**	3.58***	3.28**
Fluency	<u>2.66</u>***	2.39**	**3.13*****	2.24**	1.84**	1.88**
Coherence	<u>2.17</u>**	2.00**	**3.41*****	2.09**	1.68**	1.64**
Entertainment	2.00***	<u>2.14</u>**	**2.97****	1.81**	1.67**	1.64**
Overall	1.72***	2.00**	**2.91*****	<u>2.09</u>**	1.98**	1.89**

language models are primarily designed to output standard prose that is grammatical and sensible initially. Consequently, the addition of the PACD decoding module does not damage the overall readability of the outputs too severely. Importantly, however, we do not analyze the performance of base GPT-2 without the PACD decoding module, as the performance of GPT-2 in zero-shot scenarios is poor, therefore making this not a meaningful point of comparison. Interestingly, however, base GPT-2 with PACD (*GPT-2 -w*) outperforms the fine-tuned models either with or without the presence of the additional decoding module, suggesting that the most desirable approach may be "train *or* constrain", rather than "train *and* constrain", as this allows phoneme-level control without sacrificing grammatically.

In contrast to GPT-2, when considering Baichuan, we see that human ratings decrease when moving from standalone fine-tuned Baichuan (*Baichuan*$_{13k}$ *-w/o*) to either of the versions containing PACD. This, however, is not the case for the Articulation criteria, which increases in both PACD versions of Baichuan (i.e., *Baichuan*$_{13k}$ *-w* and *-ws*). This suggests a possible trade-off regarding the addition of constrained decoding to existing LMs, with larger models suffering from the additional control being exerted over their outputs, whereas smaller models are able to be guided towards desirable characteristics of the output language form by complimenting the reduced knowledge capacity of the smaller models during fine-tuning and zero-shot inference.

In conclusion, a significant finding from the human evaluation results is that the PACD module results in a substantial increase in the Difficulty of Articulation criteria (named "Articulation" in Table 16). This indicates the success of our constraint cascade in selecting phoneme pairs that contribute to maximal levels of phoneme-level confusion, and therefore mispronunciation (i.e., the key to a successful tongue twister).

9. Case Studies and Qualitative Analysis

9.1 Topic-to-Twister

Table 17 presents example generations using 2k and 13k training samples to investigate the different benefits and drawbacks of each model, as well as the effects of variable levels of training data in a qualitative manner. We additionally include Baichuan trained on 13k samples, and ChatGPT (GPT-3.5-Turbo) in a zero-shot manner. For these examples, we randomly select an input topic from the first 100 samples in our test set.

Table 17
Example outputs from all fine-tuned models where different training data amounts were used (2k v. 13k training examples) in addition to Baichuan with 13k training examples, and zero-shot ChatGPT in the topic-to-twister task setting. Text within square brackets indicates repeated spans, with the number of repetitions indicated next to it.

Input:	"rural brewery"
Golden:	"Rory the warrior and Roger the worrier were reared wrongly in a rural brewery."
GPT-2$_{2k}$	"In the bustling city of Chattanooga, where Chattanooga's vibrant craft beer scene thrives, The Chattanooga's craft beer scene thrives, [With Chattanooga's craft beer scene thriving, The Chattanooga's craft beer scene thrives,]***3** With Chattanooga's craft bee"
GPT-2$_{13k}$	"In the bustling town of Chattanooga, where the townspeople reside, The Chattanooga's taproom is filled with tapestries of flavors. Chattanooga's taproom is filled with the finest tapestry, From the tapestry of Chattanooga's charm, it's a sight to see. Chattanooga's taproom is filled with the finest tapestry, From the tapestry of Chattanooga's charm, it's a sight to see. Chattanooga's taproom is filled with the finest tapes"
DialoGPT$_{2k}$	"In the bustling city of Bethlehem, a bustling brewery was bustling with craftsmanship and craftsmanship was its main attraction. It was also the place to find the best beer in the city. It was also the place to find the best breweries in the city. It was also the place to find the best breweries in the country. In the bustling city, a bustling brewery was bustling with craftsmanship and craftsmanship. It was also the place to find the best beer in the city. It was als"
DialoGPT$_{13k}$	"In the bustling city of Tharrington, where the city's charm is known, There's a thriving tavern called Tharrington's, Where the taverns thrive with a vibrant atmosphere. The tavern's owner, a skilled craftsman, built a taproom that was truly grand, With a taproom full of flavors, it was truly a sight. The tavern's atmosphere was so grand, it was like a vibrant symphony, With taverns and taverns, it was truly a sigh"
BART$_{2k}$	"In the bustling city of New Zealand, a New Zealander named New Zealanders gathered at the New Zealand's New Zealand brewery, where New Zealandians gathered to discuss the New Zealander's New Zealanders' New Zealand heritage." \" \"\" "\" " " " suscept " suscept " char non"
BART$_{13k}$	"In the village of Vineland, where Vineland's vineyard thrived, Vineland and Vineland thrived. Vineland flourished, as Vineland joined the scene, As Vineland added to the scene. Vinland's vineyards, a vibrant scene, Where Vineland was the scene." " \" " \" " thwart " thwart thwarted"
Flan-T5$_{2k}$	"In the township of Westfield, a brewer named Westman brewed a brew in the townhouse, while Westfield's breweries brewed brews in the townhouses of Westfield."
Flan-T5$_{13k}$	"In the quaint town of Greenland, where gin and lager are the mainstays, Craftsmen from Greenland and Greenland gather to create a gin and lager masterpiece. The gin and gin are renowned for their exceptional craftsmanship, While the gin and gin are renowned for their exceptional taste. Greenland's gin and gin are renowned for their exceptional"
ByT5$_{2k}$	"In the village's villages, the village's villagers visit the village's villages, while Victorian villagers visit the village's villages. The village's villagers visit the village's villages, while Victorian villagers visit the village's villages. The village's villagers visit the village's villages, and the farmhouses of Victorian villages and villages in Victorian villages. The village's villagers visit the village's villages, while Victorian villagers visit the village's villages. The village's villagers visit the village's villages, all from Victorian villages, and Victorian villages are"
ByT5$_{13k}$	"In the vicinity of the village, the farmers from the farmhouse filled the villages with their farmhouses and farmsteads, while the farmers from the farmhouse farms filled the villages with their farmhouses."
Baichuan$_{13k}$	"[Breweries brew beer,]"***18**
ChatGPT	"In a rural brewery, they brewed a brew so unique, Where hops happily hopped and barley barreled sleek. With malty melodies and yeasty jigs, they thrived, But rural brewery's ruckus rhythm rarely arrived. The rural brewery's brewmaster was a friendly fellow, Fermenting fizzy flavors, flowing with yellow. With each sip, the patrons picked passionately, Pouring pints, pretty pleased, with plenty of glee. But with the twist of tongue and tricky tales, The rural brewery's name became like snails. Try as they might, lips tangled and twisted, Rural brewery remained quite disjointed and resisted. The rural brewery's brand built on rustic charm, But their tongue twister seemed to cause alarm. With each attempt, patrons tangled like vines, Rural brewery's name spinning in their minds. So, remember to take your time and don't hurry, When uttering words about a rural brewery. Though the tongue may twist and the speech may waver, The taste of their brew will make you a believer!"

9.1.1 GPT-2. First, when considering the generations from GPT-2, the generation from the 2k training sample model does not demonstrate any clear phonetic patterns, with no prominent sound repetition present. However, some orthographic repetition can be observed, with "Chattanooga", "craft", and "scene" presenting 3 ways in which the graphemes <c> and <s> appear together (as <c> is often realized as /s/, like <s>, in words such as "celery"). On the other hand, the 13k training sample generation presents a much better tongue twister, demonstrating repetition of /t/ in *"**t**own of Cha**tt**anooga[...]*

*Cha**tt**anooga's **t**aproom is filled with **t**apestries*". This is additionally complemented by the repetition of the affricate /tʃ/ in the phrase "***Ch**attanooga's **ch**arm*". Although these tongue twisters appear quite successful, it is hard to ignore the fact that in both instances the models have resorted to the repetition of very similar clauses/sentences. However, all in all, it would appear that GPT-2_{13k} has successfully generated a tongue twister that is largely grammatically coherent, demonstrates phonetic overlap, and is semantically related to the input (even if the rural nature of Chattanooga, a city in Tennessee, may be up for debate). Consequently, this first instance lends support to the proposed benefit of extended quantities of training data for the task of tongue twister generation as provided by our extension of TwistList 1.0 into TwistList 2.0, due to GPT-2_{13k} demonstrating better performance than GPT-2_{2k}. This finding also supports our claims that the proposed TwisterLister pipeline creates high-quality tongue twisters.

9.1.2 DialoGPT. Regarding DialoGPT in the topic-to-twister setting, we see high levels of redundant repetition in the 2k training sample output, such as "bustling with craftmanship and craftmanship". However, some elements are clearly tongue twister-esque, such as the initial "*In the bustling city of Bethlehem, a bustling brewery was bustling*", exploiting the voiced bilabial plosive, /b/. In the 13k training sample output, we curiously observe "Bethlehem" swapped for "Tharrington", and the exploitation of the voiceless dental fricative /θ/ also found in "thriving" and (of course) "thrive". Overall, DialoGPT demonstrates a more meaningful narrative-like generation alongside more training data, in addition to a change in primary phonemes potentially arising from the distributional properties of different phonemes across the larger training split.

9.1.3 BART. BART, on the other hand, demonstrates some less than desirable traits in the 2k setting, using extreme levels of repetition for "New Zealand" (and morphological variants) resulting in a poor quality output that only clearly represents the input topic via the inclusion of the word "breweries". Additionally, BART resorts to generating nonsense output towards the end, producing myriad punctuation and random words. In the case of 13k training samples, the output is less overtly repetitive and is more coherent (though still far from perfect), exploiting /v/ and /θ/ (a voiced labiodental fricative and a voiceless dental fricative). For example, "*In the village o**f** **V**ineland, where **V**ineland's vineyard **th**rived, **V**ineland and **V**ineland **th**rived.*" However, the generation does exhibit significant breakdown towards the end, once again producing nonsensical output (though phonetically consistent) with "thwart" and variants thereof.

9.1.4 Flan-T5. Flan-T5 generates the shortest output seen across the models for this input. The generated tongue twister in the example only appears to engage with the repetition of /b/ (rather than alongside a similar phoneme), but does so successfully, such as "*[...] a **b**rewer named Westman **b**rewed a **b**rew in the townhouse, whilst Westfield's **b**reweries **b**rewed **b**rews[...]*". However, the semantic coherence of this output is lacking, with the discussion of 2 parallel events ("Westman, who is in Westfield, brewing, whilst breweries in Westfield also brew"). Additionally, the /b/ repetition is arrived at exclusively through the exploitation of morphological variants of "brew", suggesting the signal that may have been picked up during training is on a morphological level, rather than orthographic or phonemic/phonetic (as <b> alone does not constitute a morpheme). With additional training data, the generation length changes significantly for Flan-T5. In the generation from Flan-$T5_{13k}$, there is clear repetition of the voiced velar plosive /g/ and the voiceless counterpart /k/, such as "***C**raftsmen from **G**reenland [...] **g**ather to **c**reate a **g**in and la**g**er*". However, unlike in the 2k example, the output here exhibits significant semantic redundancy as seen in phrases such as *"The gin and gin [...] while the gin and gin"*.

9.1.5 ByT5. ByT5, the largest of the models we train on numerous splits (at 582M parameters), performs poorly when considering the 2k training data split, with the output degrading into repetition. However, the repetition consists of full phrases and sentences rather than a single noun and maintains grammaticality throughout (albeit lacking in coherence). As for the version trained on 13k samples, this remedies the repetition issue. Interestingly, this version also demonstrates more clearly a phonemic pattern in the outputs, swapping between /v/ in *"**v**illager"* (and words with the same root) and /f/ in *"**f**armers"* (and words with the same root), which are voiceless/voiced counterparts of each other, and therefore an ideal pattern for a tongue twister to exploit. This is also demonstrated in the 2k version, but only weakly, with one instance of an /f/-initial word, *"**f**armhouses"*.

9.1.6 Baichuan. Baichuan, as the largest model we train (exclusively on the 13k training data split) with 7B parameters, performs rather poorly in the topic-to-twister setting. The output shown here consists exclusively of a single phrase "Breweries brew beer" repeated 18 times. Although this is a valid tongue twister in terms of relevance to the input, Baichuan is shown to very quickly get stuck in a loop, degrading the quality of the overall output. This pattern is seen frequently across other outputs from Baichuan.

9.1.7 ChatGPT. Finally, ChatGPT (GPT-3.5-Turbo) presents the longest tongue twister of the examples listed. However, this is an exception for this randomly selected generation, rather than the rule, as seen in the automatic evaluation results, where the average generation length for ChatGPT was 17.43 words. In terms of quality, ChatGPT excels at generating well-formed grammatical text, which is shown to be the case in the example generation. Additionally, the generation exploits numerous different phonemic patterns, including repetition of /h/ *"... **h**ops **h**appily **h**opped"*, /b/ *"In a rural **b**rewery they **b**rewed a **b**rew[...]"*, and /ɹ/ *"But **r**ural b**r**ewery's **r**uckus **r**hythm **r**arely a**rr**ived"*. Additionally, one demonstrated ability of ChatGPT that is not exploited by any of the other fine-tuned models (due to the fine-tuning focus) is the incorporation of additional literary techniques based on speech sounds, such as rhyme (e.g., "thr**ived**"/"arr**ived**" and "**tails**"/"sn**ails**". In addition, the coherence level of the ChatGPT output is also very significant, with the output demonstrating a clear narrative that would engage readers. Consequently, we can see why ChatGPT may score poorly on phoneme-based metrics (cf., Table 8) due to incorporating a lot of words to enhance the grammar of the tongue twister, as well as exhibiting low-level local repetition of a particular sound in a phrase, rather than maintaining the focus on the same sound for the entirety of the output.

Overall, most models appear to benefit from the presence of increased levels of training data in this example, particularly regarding producing more grammatical and coherent output. The only model for which this observation does not hold is Flan-T5, which does not have a clear change in generation quality from 2k to 13k training samples. This is in some respects to be expected, as larger models are often able to abstract away language patterns more easily than smaller models from the same amount of training data (therefore resulting in a law of diminishing returns between training data quantity and model size). However, the smaller models demonstrate the utility of our TwistList 2.0 dataset in presenting a sufficiently large increase in training data to result in tongue twister generations of demonstrably improved quality. This also shows that there is room to improve generated tongue twisters with an increase in training data exclusively before the requirement for more complex training paradigms is necessary.

9.2 Topic-to-Twister (PACD)

In contrast with the previous approach, our constrained decoding algorithm involves the random selection of an initial phoneme and nearest neighbor, through which to condition generation. Consequently, numerous tongue twisters can be generated for a single input, as long as the selected phoneme pairs differ each time. For this reason, we present an additional case study of model outputs using our PACD module in Table 18.

9.2.1 GPT-2$_{13k}$ -w/o. As discussed in §9.1, GPT-2 fine-tuned on 13k samples without the addition of our constrained decoding module demonstrates repetition of /t/ in "*town of Chattanooga[...] Chattanooga's taproom is filled with tapestries*". This is additionally complemented by the repetition of the affricate /tʃ/ in "**Ch**attanooga's **ch**arm".

9.2.2 GPT-2$_{13k}$ -w/-ws. On the other hand, fine-tuned GPT-2 with the additional constraints imposed by our PACD module demonstrates several different characteristics. First, as is the case with the non-fine-tuned model, our constrained generations are limited in length to 30 tokens, making all generations of equal length, and consequently shorter than the output of GPT-2 without these constraints. Regarding the tongue twisters, each example can be seen to be related to the input keywords, but vary between direct and abstract relations. For example, the first generation which enforced the selection of tokens starting with /b/ or /p/ demonstrates a clear direct reference to the input word "brewery", something which has been afforded to the model as the selected initial phonemes match one of the word-initial phonemes of the input (which is the approach we take in §8.2).

Lastly, the generations exploiting /ɹ/ and /w/ are similarly abstract (but perhaps to a lesser extent), referring to relevant words such as "water", "wine", and "wood working" (the latter activity being more expected in a rural locale). Regarding grammatically, the /b, p/ example is largely grammatical (if "boys" had a possessive apostrophe, to create "the bar boys' breweries" as a noun phrase") but is cut off at 30 tokens, resulting in an incomplete sentence. However, semantically it is hard to follow due to the high level of temporal adverbs used. Likewise, the /ɹ, w/ generation is mostly grammatical, with the presence of numerous relative clauses leading to difficulty in understanding (however, this does require some liberty in accepting that "wheat wine wines" and "win ware warehouses" are allowable compound nouns). With the subword loop turned on for GPT-2 using PACD, we see similar outputs in style to the full-word version. However, one difference we observe is a slight reduction in grammaticality towards the end of the /b, p/ version, with the phrase "and bro pal put bulls before", which is difficult to parse. Similar effects can be seen with the /ɹ, w/ version, where the outputs are similar in word choice (as expected, given some whole words will form the stem of longer words in the subword version) and the overall result is perhaps slightly more difficult to parse grammatically.

9.2.3 GPT-2 -w/-ws. Regarding non-fine-tuned base GPT-2 with the addition of our module, all generations demonstrate similar levels of grammaticality to those seen in our fine-tuned model. On the other hand, the generations can be considered less literary, with the /b, p/ example suggesting the model is generating instructional text in "*But before you do begin building your own brewery[...]*". Similarly to previously discussed

Table 18
Example outputs from various models using our novel constrained decoding algorithm for tongue twister generation. -w denotes models using our PACD decoding algorithm on full words only, whilst -ws indicates the subword version of PACD. We present 2 generations for each model using our PACD decoding approach. Each example exploits the phonemes /b, p/ and /ɹ, w/, respectively. To achieve this, we overwrite the initial phoneme selection process of ph_1 in the PACD algorithm in order to present diverse outputs from the same input (as /b, p/ would be selected exclusively otherwise).

Input:	"rural brewery"
Golden:	"Rory the warrior and Roger the worrier were reared wrongly in a rural brewery."
GPT-2$_{13k}$	"In the bustling town of Chattanooga, where the townspeople reside, The Chattanooga's taproom is filled with tapestries of flavors. Chattanooga's taproom is filled with the finest tapestry, From the tapestry of Chattanooga's charm, it's a sight to see. Chattanooga's taproom is filled with the finest tapestry, From the tapestry of Chattanooga's charm, it's a sight to see. Chattanooga's taproom is filled with the finest tapes"'
GPT-2$_{13k}$ -w	"Brewers brew beer in the barn brewing process before brewers bring their beers to the bar boys breweries prior to being brewed beforehand but before being bottled previously because of"
	"With their rust wood working wonders with their water works while watering wheat wine wines within their win ware warehouses where they are waiting for the weather to warm up"
GPT-2$_{13k}$ -ws	"Brewers brew beer in the breweries belonging to both baker boys and bar patrons but the bartender prefers the beers brewed by bud brothers and bro pal pit bulls before"
	"With a rust wood wind ward rip weed from the woods while watering the water works well within the walls of the warehouse where the work room is wall rolling"
GPT-2 -w	"But before you do begin building your own brewery please bear in place that you are not a brewer but a person who has been brewing for a bit because "
	"What is the reason for this one with the word rural rather than regional where it is written when it is referring to the region of the world where the"
GPT-2 -ws	"But I believe that the best place to begin is with the basics before you proceed beyond that point because the more you practice brewing beer properly both in the"
	"What is the reason for this request with the word rural when you are referring to a region that is not a regionally relevant one where there is a real"
Baichuan$_{13k}$	"[Breweries brew beer]"***18**"
Baichuan$_{13k}$ -w	"Try to drink tea in the town during dinner time drinking delicious drinks from the tap tasting different types of tasty treats trying to taste the differ den between two"
	"What is the relations with rural roads running through resident rial ways within regional regions where rivers run through rocky rid rig road way runs through region rich river routes"
Baichuan$_{13k}$ -ws	"Try to drink tea in the township of taverna towns during the daytime drinking tour dates downtown drinker drinks dry den dutch town tavern a day time tasting tours tourist"
	"What is the relationship between the rural region and the regional railway route running through rut rum runners runs through rocky regions with rich resources while retaining its rustic roots"
Baichuan -w	"Try to twist the tongue as tight together as time does during the day today due to the temper ter differ ten times too different degrees depending on the direction"
	"Write a one word response to each of the words below with a relevant rural reference within the reply which is not a repeated ref rain water recycling wind renewable"
Baichuan -ws	"Try to twist the tongue as tightly as desired during the time of delivery trying to deliver the twistiest tong toe track ter day till the target is delivered tight"
	"Write a one word response to each of the words below with a relevant rural reference within the reply which is not a repeated ref rain water recycling rainwater runoff"

generations with the PACD module, all of these generations are hindered by the 30-token limit, with examples ending in "because", "his desire to try" (which suggests the verb "try" will take an additional argument) and "the". Overall, similar results are seen with the subword loop enabled (-ws), resulting in variations of the same output.

9.2.4 Baichuan$_{13k}$. As discussed previously, fine-tuned Baichuan without the addition of PACD produces a valid 3-word tongue twister phrase that achieves alliteration of /b/, but quickly falls into the degenerate pattern of repeating this phrase 18 times, rather than continuing to extend the tongue twister in a unique and entertaining fashion.

9.2.5 Baichuan$_{13k}$ -w/-ws. With the addition of full-word PACD to the fine-tuned version of Baichuan, we successfully avoid the repetition trap (due to repetition restrictions at decoding time). For the first example, the enforcement of /t, d/ is evident in grammatical phrases such as *"**T**ry **t**o **d**rink the **t**ea in the **t**own **d**uring **d**inner **t**ime..."*, and is continued throughout. Overall, the first generation is grammatical, only demonstrating a clear degradation towards the end with the generation of "differ den", which hinders fluency and coherence. On the other hand, in the /ɹ, w/ example, the sound overlap is demonstrated, but the coherence and fluency of the output are shown to suffer much earlier, with sequences such as *"rivers run through rocky rid rig road ways runs through region rich river routes"*. An overall pattern can be seen that generation quality begins to drop after a single suboptimal token is selected due to the detrimental impact it has on the following token's prediction. When comparing to the subword enabled PACD (-ws) we see the initial difference in the 7th word of the /t, d/ example, where "town" has been extended to "township", resulting in the remainder of the generation diverging from that of the full-word version of PACD, which is similarly reflected in the /ɹ, w/ example as "relation" becomes "relationship". Overall, due to the constraints in place, both present difficult-to-articulate sequences, with the overall quality between subword and full-world versions being minimal and subjective.

9.2.6 Baichuan -w/-ws. Regarding base Baichuan with the addition of PACD, we see a lack of topical relevance in the /t, d/ examples for both full-word and subword versions, with the input of "rural brewery" not being reflected in the semantics of the output (although the generation has noted the request for a tongue twister in the starting stem, and references it directly). This is also seen in the /ɹ, w/ versions but to a lesser extent, as the topic word "rural" starts with an allowable phoneme, /ɹ/, causing it to generate in the phrase "**r**ural **r**eference". In this example, therefore, we see that fine-tuned Baichuan is better able to understand the desired topic (due to being trained on the topic-to-twister setting), while the base model produces overtly generic text, but with repetitive phonology.

Overall, the examples demonstrate the effectiveness of the simple constrained decoding module that constitutes PACD. In the examples generated through this module, the ability to control token repetition means we avoid the pitfalls of standard fine-tuned models that frequently default to phrasal level repetition, ensuring that tongue twisters generated with PACD use a wide range of vocabulary. Additionally, due to the constraints being phoneme-based, we can exercise control over the present sounds even when permitting the generation of a small set of function words to facilitate grammatical correctness. To this end, similar to the pipeline of TwisterLister (§4.1), PACD allows us to ensure phoneme overlap within the output, rather than relying on this signal being picked up during fine-tuning. The case studies additionally demonstrate that rule-based decoding of a pre-trained language model can result in tongue twisters that are comparable to those of fine-tuned models, and that future work in the area should make extensive use of constrained decoding approaches such as PACD, rather than exclusively fine-tuning on input-output pairs.

9.3 Style-Transfer

We now qualitatively investigate the generations from our additional task setting using style-transfer. Table 19 presents example generations across 2k training samples and

Table 19
Example outputs from all fine-tuned models where different training data amounts were used (2k v. 13k) in addition to Baichuan with 13k training examples, and zero-shot ChatGPT in the style-transfer task setting. Please note that the mass majority of ChatGPT outputs resemble tongue twisters, although this randomly selected input topic generated twisted syntax in this particular case.

Input:	"The small crimson truck traveled along Limuru street."
Golden:	"The little red lorry went down Limuru road."
GPT-2$_{2k}$	"The small carriages of the small carriages, with their small wheels, were moving along Limuru's road. "The small yellow truck traveled along Limuru's road. The yellow cars of the yellow cars, with their yellow wheels, were traveling along Limuru's road." "The yellow yellow truck traveled along Limuru's road." The yellow cars of the yellow cars, with their yellow wheels, were traveling along Limuru's road.""The yellow yellow truck traveled along Limuru's road "
GPT-2$_{13k}$	"The small red truck, with its small red tires, was a sight to see. The small red truck, with its small red tires, was a sight to see. The small red truck, with its small red tires, was a sight to see. The small red truck, with its small red tires, was a sight to see. The small red truck, with its small red tires, was a sight to see. The small red truck, with its small red tires, was a sight"
DialoGPT$_{2k}$	"It passed through the town of Limburg, where the small town of Limburg's inhabitants were located. It passed through the town of Limburg, where the town of Limburg's inhabitants were located. It passed through the town of Limburg, where the town of Limburg's inhabitants were located. It passed through the town of Limburg, where the town of Limburg's inhabitants were located. It passed through the town of Limburg, where the town of Limburg's inhabitants were located"
DialoGPT$_{13k}$	"It passed by the small houses and the small houses' small houses, while the small houses' small houses and the small houses' small houses were observed."The minuscule minivan passed by the mansions and mansions' mansions, while the mansions' mansions and the mansions' mansions were seen." Improved version: The minuscule minivan passed by the mansions and mansions' mansions, while the mansions' mansions and the mansions"
BART$_{2k}$	"The limo journeyed along Limuru thoroughfare." " churn " churned B hollow f un ümbled inc ops op edy - standard r unilaterally worsened thumb pl thrust res ed churn"
BART$_{13k}$	""The little red limousine traveled down Limuru thoroughfare." \" \" \" \"\" \" \"\" \" \uff \uff \" \" \" L " t \", ' churn M L \" " uff \churn"
Flan-T5$_{2k}$	"The yellow limo limped along Limuru limoway."
Flan-T5$_{13k}$	"The little lilac lorry limped down Limuru lanes."
ByT5$_{2k}$	"The little limuru truck looked along Limuru street."
Byt5$_{13k}$	"The little red truck wandered along Limuru thoroughfare."
Baichuan$_{13k}$	"Limuru's little red truck drove along Limuru road."
ChatGPT	"Truck crimson, small, along Limuru street did travel."

13k training samples in order to investigate the different benefits and drawbacks of each model, as well as the effects of variable levels of training data in a qualitative manner. We further include Baichuan trained on 13k samples, and zero-shot ChatGPT (GPT-3.5-Turbo).

As demonstrated in the *Input* and *Gold Standard* fields, the generated paraphrase is not easily identifiable as a tongue twister and is, therefore, a prime example of a successful paraphrase to enable style-transfer (as the twisted style has been removed in order to have parallel data to aid in identifying the desired style to transfer from the source to the target), while the gold standard reference from which the paraphrase was derived exploits the common articulation difficulty and confusion between the rhotic /ɹ/ and lateral /l/ (Georgievska-Jancheska 2019), encouraging metathesis (the accidental transposition of speech sounds across words).

9.3.1 GPT-2. First, the main element to notice with either generation from GPT-2 is the excessive length. This is particularly startling when you consider the nature of style-transfer, where the ideal tongue twister version of a text would be of similar length to the original. However, upon further inspection, it is clear that GPT-2 resorts to blatant repetition of very similar content to constitute the delta in length. Consequently, if we only take the first sentence of the 2k version as the output *"The small carriages of small carriages, with their small wheels, were moving along Limuru's road."*, the output can more clearly be seen as a style-transferred version of the input. However, this generation is still poor, with little clear phoneme-level repetition (excluding repetition of the same word, such as "small" in the quoted passage) and much semantic redundancy. Regarding the 13k training example output, GPT-2 yet again exhibits large levels of repetition, but in this case, the repetition is purer, being a verbatim repetition of the initial sentence numerous times. In addition, the output is free of any grammatical errors. In this example, there is clear use of the sibilant /s/ in *"The **s**mall red truck with it**s** **s**mall red tires, was a **s**ight to **s**ee"*. Additionally, the input semantics of a small truck have been maintained, but the location (Limuru, a town in Kenya) has been lost.

9.3.2 DialoGPT. On the other hand, the 2k version of the dialogue fine-tuned GPT-2, DialoGPT lacks semantic coherence due to using deixis in the form of "it", which is not resolved as a cataphoric reference later in the text, making "it" ambiguous. A potential attempt at phoneme repetition is present with "Limburg" and "located", but this is tenuous. Overall, the generation with 2k training examples does not clearly present a tongue twister. On the other hand, the extended 13k example demonstrably resembles a tongue twister, achieving repetition of /s/ (e.g., "*__S__mall houses and the **s**mall houses' **s**mall houses'*) and /m/ (e.g., *'The **m**inuscule **m**inivan passed by the **m**ansion and the **m**ansions' **m**ansions*"). However, this generation also exhibits a strange structure, including "improved version" as part of the output. If only a subset of this generation is considered, *"The minuscule minivan passed by the mansion and the mansions' mansions"*, the output constitutes a high-quality tongue twister. Although it may be semantically bizarre, to discuss the ownership of mansions by other mansions, it is not grammatically invalid, and human-authored tongue twisters also frequently convey unusual semantics to increase their strange nature.

9.3.3 BART. Unlike the GPT-2–based models, BART produces style-transferred versions of the input that are much closer in length to the original input, even without needing to exclude sentence-level repetition. In the 2k example, the initial sentence *"The limo journeyed along Limuru thoroughfare"* is a grammatically and semantically valid output, but does not resemble a tongue twister outside of the lateral /l/ in both "limo" and "Limuru". However, the sentence following this consists primarily of noise. On the other hand, alongside the increase in training data comes an improvement in tongue twister quality, with *"The **l**itt**l**e **r**ed **l**imousine trave**l**ed down **L**imuru thoroughfare"* appearing to exploit the articulatory similarity between /l/ and /ɹ/ without relying on single-word repetition. Again, however, the output devolves into noise towards the end, consisting of various punctuation and subwords.

9.3.4 Flan-T5. Flan-T5 (alongside ByT5) presents the most clear paraphrases of the desired output, containing no unnecessary repetition or noise. In the 2k example, Flan-T5 repeats similar phonemes, exploiting the similar phonetic categories, laterals and glides (or semivowels), of which /l/ and /j/ belong to, respectively, *"The **y**e**ll**ow **l**imo **l**imped a**l**ong **L**imuru **l**imoway."*. On the other hand, from more training examples, Flan-T5 presents a paraphrase that is more faithful to the original input, relying on the repetition of /l/ exclusively: *"The **l**itt**l**e **l**i**l**ac **l**orry **l**imped down **L**imuru **l**anes."* (assuming we take "lorry" to be more semantically related to "truck" than "limousine" is).

9.3.5 ByT5. ByT5 shows equivalent performance to Flan-T5 in this example, opting to alliterate different parts of speech to Flan-T5, but overall having a similar effect in the 2k example. On the other hand, the 13k version doesn't present alliteration as clearly as Flan-T5, but does alternate between related phonemes, /l, ɹ, w/, in "**l**ittle **r**ed truck **w**andered".

9.3.6 Baichuan. In the style-transfer task setting, Baichuan can be seen to perform much better than witnessed in the previous topic-to-twister task setting. Similar to other generations, Baichuan exploits the /ɹ/ and /l/ similarity with *"**L**imuru's **l**ittle **r**ed t**r**uck d**r**ove a**l**ong **L**imuru **r**oad"*, where every word of the output contains one of these sounds, and alternation is frequent.

9.3.7 ChatGPT. Finally, ChatGPT in a zero-shot setting appears to strongly misinterpret the given prompt, outputting non-standard syntax such as swapping the order of an adjective and noun in "truck crimson" (reminiscent of the speaking style of Star Wars' Yoda). Additionally, the ChatGPT generation does not appear to reflect a tongue twister in any clear sense. It is important to note, however, that this is not a common pattern with ChatGPT outputs, but is the case for the randomly selected case study example.

Overall, as with the topic-to-twister setting, we can see a clear benefit in the style-transfer task formulation of using more training data, with all models producing better quality outputs both in terms of phonetic patterns, grammatical validity, and semantic coherence. An overarching theme, however, is that several models misinterpret the desire to paraphrase a single sentence as a tongue twister, instead producing outputs that are very long. However, if edited in post (often simply by taking the first sentence), these generations are often still valid (if not perfect).

10. Conclusion

In this article, we have presented multiple novel contributions towards the development of more phonetically and phonologically aware models for the task of tongue twister generation. We presented a pipeline for the generation of tongue twisters at scale using large language models that encourage uniqueness and non-derivative examples through the careful selection of a candidate vocabulary (TwisterLister) to develop a large dataset of machine-generated tongue twisters (TwistList 2.0). We then fine-tuned a series of smaller language models on the resulting dataset and observed that the topic-to-twister and style-transfer task settings for tongue twister generation exhibit different characteristics when considering the benefit of additional training data, as well as observing that different models perform differently in the two task settings. These findings demonstrate a fundamental difference in the requirements of the two approaches and further motivated the need for additional training data as supplied by TwistList 2.0 through the use of the TwisterLister pipeline. We additionally presented a novel algorithm (PACD) that implements hard lexical constraints based on the phonemic characteristics of words that can be used to realize tongue twisters from a causal autoregressive language model by accessing the next token predictions and applying a cascade of filters. We then extensively evaluated the generations from our proposed approaches, presenting both automatic and human evaluation. In the former, we additionally presented 2 novel metrics for measuring the sound complexity of a generated tongue twister based on the concept of phoneme edit distance (iPED and oPED). With these fine-tuned models and constrained decoding module, we then provided an in-depth exploration of the generation characteristics through case studies investigating the propensity of each model to generate high-quality tongue twisters, as well as the differences seen via the addition of more training data (to move away from reliance on automatic metrics that do not reveal fundamental qualitative differences). Overall, we find that straight fine-tuning of language models for a tongue twister generation task still has substantial room for improvement to meet human-authored standards. However, we additionally demonstrate that simple constrained decoding approaches are able to generate better tongue twisters than only fine-tuned models, particularly due to the phoneme-level awareness which allows for more difficult-to-articulate sound combinations to be present in an output. We additionally envision the techniques and approaches presented herein to be beneficial to the creative NLG community, particularly in regard to the generation of phonemically conditioned language forms (e.g., poetry, lyrics, and puns) to explore methods of constraining token outputs while simultaneously taking advantage of the power of modern LLMs.

We hope to witness increased interest in the area of tongue twister generation, as well as other niche areas of creative language generation that pique the interest of newcomers to the NLG domain, as well as people from wider domains such as literature and (non-computational) linguistics, and the general public (where creative language generation may offer a more accessible and intriguing entry into the NLP and machine learning communities). In furthering work in this area, we believe reinforcement learning approaches may prove fruitful, in addition to incorporating a differentiable version of something akin to phonemic edit distance as an additional loss function to optimize. Furthermore, although we present new metrics (oPED/iPED), we encourage the development of more robust general metrics for tongue twisters and other forms of creative language that can adequately balance the requirements of being grammatical as well as exhibiting significant levels of sound repetition.

Appendix A. Evaluation Rubric

Below is the evaluation rubric presented to human evaluators. The prompts have the following format: ``**[criterion description]**\n Instruction: Generate a tongue twister relating to **[input]**\n Response: **[output]**\n Rate the response from 1 to 5:\n **[rubric]**'' Where *criterion description* refers to the first line of the examples below (e.g., "Is the model proficient in..."), *[input]* is the input to the model (either a topic in the topic-to-twister setting, or standard non-literary text in the style-transfer setting), *[output]* is an LM's response to a given input topic/text, and *[rubric]* is the 5-point rating system as outlined below.

Relevance

Is the model proficient in generating text that is relevant to the input topic?

1. The model completely ignores the input topic and generates irrelevant text.
2. The model generates text that is mostly irrelevant to the input topic, with minimal and unclear association.
3. The model generates text that is partially relevant to the input topic, but the association is weak and inconsistent.

4. The model generates text that is mostly relevant to the input topic, with clear association but occasional lapses.
5. The model excels in generating relevant text, where the responses are consistently on topic and the association is clear.

Difficulty of Articulation

Is the model proficient in generating text that is difficult to pronounce due to alliteration and phonetic complexity?

1. The model generates text that is no harder to pronounce and articulate than standard writing.
2. The model generates text that is slightly more challenging to say than standard writing, demonstrating some simple techniques such as alliteration.
3. The model generates text that is somewhat difficult to say but clearly exhibits techniques such as alliteration.
4. The model generates text that is generally difficult to say, with techniques such as alliteration, but also alternating between similar sounds.
5. The model generates text that is highly phonetically complex and difficult to say, consistently exploiting repetition of closely related sounds and alliteration.

Fluency

Is the model proficient in generating grammatical and well-formed text?

1. The model produces text with no grammatical phrases or spans.
2. The model generates text that is largely ungrammatical but with some grammatically valid sequences.
3. The model generates text that contains grammatically valid sequences but overall is difficult to parse.
4. The model generates text that is generally grammatical and well-formed, with minor errors or awkward phrasing.
5. The model produces text that is grammatically correct and well-formed, demonstrating a strong command of the English language.

Coherence

Is the model proficient in generating semantically coherent and logical text?

1. The model neglects to generate semantically coherent text, producing text that is nonsensical in meaning.

2. The model generates text that is mostly incoherent, with only occasional hints of logical meaning.
3. The model generates text that is partially coherent, but the text lacks logical structure and consistency.
4. The model generates text that is generally coherent, with a logical structure and clear meaning, though minor inconsistencies may be present.
5. The model excels in generating semantically coherent text, where the text is logically structured and maintains a clear and consistent meaning.

Entertainment

Is the model proficient in generating text that a human reader would find entertaining or amusing?

1. The model demonstrates no creativity, either in the content or the structure of the text, resulting in uninteresting or unamusing outputs.
2. The model generates text with minimal creativity, resulting in outputs that are only slightly interesting or amusing.
3. The model generates text that is somewhat entertaining or amusing, but the creativity in content and structure is limited.
4. The model generates text that is generally entertaining and amusing, showing noticeable creativity in content and structure, though some outputs may be less engaging.
5. The model excels in creating entertaining and amusing text, demonstrating creativity in both content and structure and consistently producing engaging and enjoyable outputs.

Overall Quality

Is the model proficient in generating high quality English tongue twisters?

1. The model fails to generate high quality text that is recognizable as a tongue twister.
2. The model generates text that slightly resembles a tongue twister.
3. The model generates text that is recognizable as a tongue twister, but is lacking in fluency, coherence, or entertainment value.
4. The model generates text that is easily recognizable as a tongue twister, but may fall short of high quality.
5. The model excels in generating texts that resemble high-quality tongue twisters that are difficult to pronounce, make sense, and are entertaining.

Acknowledgments

Tyler Loakman is supported by the Centre for Doctoral Training in Speech and Language Technologies (SLT) and their Applications funded by UK Research and Innovation [grant number EP/S023062/1]. Chen Tang is supported by the China Scholarship Council (CSC) for his doctoral study (File No. 202006120039).

References

Anil, Rohan, Sebastian Borgeaud, Yonghui Wu, Jean-Baptiste Alayrac, Jiahui Yu, et al. 2023. Gemini: A family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*.

Askari, Arian, Mohammad Aliannejadi, Chuan Meng, Evangelos Kanoulas, and Suzan Verberne. 2023. Expand, highlight, generate: RL-driven document generation for passage reranking. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 10087–10099. https://doi.org/10.18653/v1/2023.emnlp-main.623

Brown, Tom B., Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, et al. 2020. Language models are few-shot learners. *CoRR*, abs/2005.14165.

Buciluă, Cristian, Rich Caruana, and Alexandru Niculescu-Mizil. 2006. Model compression. In *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '06, pages 535–541. https://doi.org/10.1145/1150402.1150464

Chakrabarty, Tuhin, Philippe Laban, Divyansh Agarwal, Smaranda Muresan, and Chien-Sheng Wu. 2023. Art or artifice? Large language models and the false promise of creativity. *arXiv preprint arXiv:2309.14556*. https://doi.org/10.1145/3613904.3642731

Chall, Jeanne Sternlicht and Edgar Dale. 1995. *Readability Revisited: The New Dale-Chall Readability Formula*. Brookline Books.

Chang, Yongzhu, Rongsheng Zhang, Lin Jiang, Qihang Chen, Le Zhang, and Jiashu Pu. 2023. Sudowoodo: A Chinese lyric imitation system with source lyrics. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 99–105. https://doi.org/10.18653/v1/2023.emnlp-demo.8

Chen, Hong, Raphael Shu, Hiroya Takamura, and Hideki Nakayama. 2021. GraphPlan: Story generation by planning with event graph. In *Proceedings of the 14th International Conference on Natural Language Generation*, pages 377–386. https://doi.org/10.18653/v1/2021.inlg-1.42

Chiang, Cheng Han and Hung-yi Lee. 2023. Are synonym substitution attacks really synonym substitution attacks? In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 1853–1878. https://doi.org/10.18653/v1/2023.findings-acl.117

Chung, Hyung Won, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Yunxuan Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, Albert Webson, Shixiang Shane Gu, Zhuyun Dai, et al. 2022. Scaling instruction-finetuned language models. *arXiv preprint arXiv:2210.11416*.

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)*, pages 7282–7296. https://doi.org/10.18653/v1/2021.acl-long.565

Clements, G. Nick and Rachid Ridouane. 2011. *Where Do Phonological Features Come From?: Cognitive, Physical and Developmental Bases of Distinctive Speech Categories*, 1st ed. Language Faculty and Beyond Series. John Benjamins Publishing Company, Amsterdam/Philadelphia. https://doi.org/10.1075/lfab.6

De Lacy, Paul V. 2007. *The Cambridge Handbook of Phonology / [electronic resource]*. Cambridge University Press, Cambridge. https://doi.org/10.1017/CBO9780511486371

Devlin, Jacob, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. 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 4171–4186.

Flesch, R. R. 1948. A new readability yardstick. *Journal of Applied Psychology*, 32:2211–2223. https://doi.org/10.1037/h0057532, PubMed: 18867058

Foster, Mary Ellen and Michael White. 2007. Avoiding repetition in generated text. In *Proceedings of the Eleventh European Workshop on Natural Language Generation (ENLG 07)*, pages 33–40. https://doi.org/10.3115/1610163.1610170

Franceschelli, Giorgio and Mirco Musolesi. 2023. On the creativity of large language models. *arXiv preprint arXiv:2304.00008*. https://doi.org/10.1007/s00146-024-02127-3

Francis, W. Nelson and Henry Kucera. 1979. The Brown corpus. *Department of Linguistics, Brown University*.

Geisel, Theodore Seuss. 1965. *Fox in Socks: Dr. Seuss's Book of Tongue Tanglers*. Random House.

Georgievska-Jancheska, Tatjana. 2019. Lambdacism, rhotacism and sigmatism in preschool children: Frequency and distribution. *Open Access Macedonian Journal of Medical Science*, 7(3):336–340. https://doi.org/10.3889/oamjms.2019.144, PubMed: 30833997

Gick, Bryan, Ian Wilson, and Donald Derrick. 2013. *Articulatory Phonetics*. John Wiley & Sons.

Gómez-Rodríguez, Carlos and Paul Williams. 2023. A confederacy of models: A comprehensive evaluation of LLMs on creative writing. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 14504–14528. https://doi.org/10.18653/v1/2023.findings-emnlp.966

Guerini, Marco, Gözde Özbal, and Carlo Strapparava. 2015. Echoes of persuasion: The effect of euphony in persuasive communication. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1483–1493. https://doi.org/10.3115/v1/N15-1172

Gunning, R. 1971. *The Technique of Clear Writing*. McGraw-Hill.

Gupta, Manish and Puneet Agrawal. 2022. Compression of deep learning models for text: A survey. *ACM Transactions on Knowledge Discovery from Data*, 16(4):Art. 61. https://doi.org/10.1145/3487045

Gupta, Prakhar, Shikib Mehri, Tiancheng Zhao, Amy Pavel, Maxine Eskenazi, and Jeffrey Bigham. 2019. Investigating evaluation of open-domain dialogue systems with human generated multiple references. In *Proceedings of the 20th Annual SIGdial Meeting on Discourse and Dialogue*, pages 379–391. https://doi.org/10.18653/v1/W19-5944

Hinton, Geoffrey, Oriol Vinyals, and Jeffrey Dean. 2015. Distilling the knowledge in a neural network. In *NIPS Deep Learning and Representation Learning Workshop. arXiv preprint arXiv:1503.02531*.

Hokamp, Chris and Qun Liu. 2017. Lexically constrained decoding for sequence generation using grid beam search. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1535–1546. https://doi.org/10.18653/v1/P17-1141

Hong, Xudong, Asad Sayeed, Khushboo Mehra, Vera Demberg, and Bernt Schiele. 2023. Visual writing prompts: Character-grounded story generation with curated image sequences. *Transactions of the Association for Computational Linguistics*, 11:565–581. https://doi.org/10.1162/tacl_a_00553

Hu, Edward J., Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. LoRA: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*.

Hu, Edward J., Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*.

Iso, Hayate. 2022. Autotemplate: A simple recipe for lexically constrained text generation. In *Proceedings of the 17th International Natural Language Generation Conference*.

Jessen, Michael. 2008. Forensic phonetics. *Language and Linguistics Compass*, 2(4):671–711. https://doi.org/10.1111/j.1749-818X.2008.00066.x

Keh, Sedrick Scott, Steven Y. Feng, Varun Gangal, Malihe Alikhani, and Eduard Hovy. 2023. PANCETTA: Phoneme aware neural completion to elicit tongue twisters automatically. In *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics*, pages 491–504. https://doi.org/10.18653/v1/2023.eacl-main.36

Kember, Heather, Kathryn Connaghan, and Rupal Patel. 2017. Inducing speech errors in dysarthria using tongue twisters. *International Journal of Language & Communication Disorders*, 52(4):469–478.

https://doi.org/10.1111/1460-6984.12285, PubMed: 27891744

Kingma, Diederik P. and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Klausenburger, Jürgen. 1970. *French Prosodics and Phonotactics: An Historical Typology*, 1st ed., Beihefte Zur Zeitschrift Für Romanische Philologie Series. Walter de Gruyter GmbH, Tübingen. https://doi.org/10.1515/9783111328119

Ladefoged, Peter. 1996. *Elements of Acoustic Phonetics*, University of Chicago Press. https://doi.org/10.7208/chicago/9780226191010.001.0001

Lewis, Mike, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880. https://doi.org/10.18653/v1/2020.acl-main.703

Li, Yucheng, Frank Guerin, and Chenghua Lin. 2022. The secret of metaphor on expressing stronger emotion. In *Proceedings of the 3rd Workshop on Figurative Language Processing (FLP)*, pages 39–43. https://doi.org/10.18653/v1/2022.flp-1.6

Li, Yucheng, Shun Wang, Chenghua Lin, Frank Guerin, and Loic Barrault. 2023a. FrameBERT: Conceptual metaphor detection with frame embedding learning. In *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics*, pages 1558–1563. https://doi.org/10.18653/v1/2023.eacl-main.114

Li, Zhuoyan, Hangxiao Zhu, Zhuoran Lu, and Ming Yin. 2023b. Synthetic data generation with large language models for text classification: Potential and limitations. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 10443–10461. https://doi.org/10.18653/v1/2023.emnlp-main.647

Li, Yizhi, Ruibin Yuan, Ge Zhang, Yinghao Ma, Xingran Chen, Hanzhi Yin, Chenghao Xiao, Chenghua Lin, Anton Ragni, Emmanouil Benetos, et al. 2024. MERT: Acoustic music understanding model with large-scale self-supervised training. In *Proceedings of the 12th International Conference on Learning Representations (ICLR)*.

Lin, Chin Yew. 2004. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81.

Loakman, Tyler, Aaron Maladry, and Chenghua Lin. 2023. The iron(ic) melting pot: Reviewing human evaluation in humour, irony and sarcasm generation. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 6676–6689. https://doi.org/10.18653/v1/2023.findings-emnlp.444

Loakman, Tyler, Chen Tang, and Chenghua Lin. 2023. TwistList: Resources and baselines for tongue twister generation. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 579–589. https://doi.org/10.18653/v1/2023.acl-short.51

Lu, Ximing, Sean Welleck, Peter West, Liwei Jiang, Jungo Kasai, Daniel Khashabi, Ronan Le Bras, Lianhui Qin, Youngjae Yu, Rowan Zellers, Noah A. Smith, and Yejin Choi. 2022. NeuroLogic A*esque decoding: Constrained text generation with lookahead heuristics. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 780–799. https://doi.org/10.18653/v1/2022.naacl-main.57

Manjavacas, Enrique, Mike Kestemont, and Folgert Karsdorp. 2019. Generation of hip-hop lyrics with hierarchical modeling and conditional templates. In *Proceedings of the 12th International Conference on Natural Language Generation*, pages 301–310. https://doi.org/10.18653/v1/W19-8638

McCutchen, Deborah and Charles A. Perfetti. 1982. The visual tongue-twister effect: Phonological activation in silent reading. *Journal of Verbal Learning and Verbal Behavior*, 21(6):672–687. https://doi.org/10.1016/S0022-5371(82)90870-2

Mortensen, David R., Patrick Littell, Akash Bharadwaj, Kartik Goyal, Chris Dyer, and Lori Levin. 2016. PanPhon: A resource for mapping IPA segments to articulatory feature vectors. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 3475–3484.

O'Halloran, Ken D. 2020. A tongue-twister to translation? Increased complexity of genioglossus movement during wakefulness in persons with obstructive sleep apnoea. *The Journal of Physiology*, 598(3):435–436. https://doi.org/10.1113/JP279382, PubMed: 31860741

OpenAI, Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, et al. 2023. GPT-4 technical report.

Ouyang, Long, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. 2022. Training language models to follow instructions with human feedback. *arXiv preprint arXiv.2203.02155*.

Papineni, Kishore, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: A method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318. https://doi.org/10.3115/1073083.1073135

Ploujnikov, Artem and Mirco Ravanelli. 2022. SoundChoice: Grapheme-to-phoneme models with semantic disambiguation. In *Proceedings of Interspeech 2022*, pages 486–490. https://doi.org/10.21437/Interspeech.2022-11066

Popescu-Belis, Andrei, Àlex R. Atrio, Bastien Bernath, Etienne Boisson, Teo Ferrari, Xavier Theimer-Lienhard, and Giorgos Vernikos. 2023. GPoeT: A language model trained for rhyme generation on synthetic data. In *Proceedings of the 7th Joint SIGHUM Workshop on Computational Linguistics for Cultural Heritage, Social Sciences, Humanities and Literature*, pages 10–20. https://doi.org/10.18653/v1/2023.latechclfl-1.2

Potash, Peter, Alexey Romanov, and Anna Rumshisky. 2018. Evaluating creative language generation: The case of rap lyric ghostwriting. In *Proceedings of the Second Workshop on Stylistic Variation*, pages 29–38. https://doi.org/10.18653/v1/W18-1604

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

Raffel, Colin, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67.

Rose, Stuart, Dave Engel, Nick Cramer, and Wendy Cowley. 2010. Automatic keyword extraction from individual documents. In *Text Mining: Applications and Theory*, pages 1–20. https://doi.org/10.1002/9780470689646.ch1

Roush, Allen, Sanjay Basu, Akshay Moorthy, and Dmitry Dubovoy. 2022. Most language models can be poets too: An AI writing assistant and constrained text generation studio. In *Proceedings of the Second Workshop on When Creative AI Meets Conversational AI*, pages 9–15.

Smith, E. A. and R. J. Senter. 1967. Automated readability index. *AMRL-TR. Aerospace Medical Research Laboratories (6570th)*. pages 1–14.

Somoff, Victoria. 2014. Four is not fourteen: Tongue twister patterns and the unmastery of language. *Western Folklore*, 73(2/3):195–215.

Sugiharto, Prasetyawan, Yan Santoso, and Maila Shofyana. 2022. Teaching English pronunciation using tongue twister. *Acitya: Journal of Teaching and Education*, 4(1):189–197. https://doi.org/10.30650/ajte.v4i1.3210

Sun, Jiao, Anjali Narayan-Chen, Shereen Oraby, Shuyang Gao, Tagyoung Chung, Jing Huang, Yang Liu, and Nanyun Peng. 2022. Context-situated pun generation. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 4635–4648. https://doi.org/10.18653/v1/2022.emnlp-main.306

Tang, Chen, Chenghua Lin, Henglin Huang, Frank Guerin, and Zhihao Zhang. 2022. EtriCA: Event-triggered context-aware story generation augmented by cross attention. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 5504–5518. https://doi.org/10.18653/v1/2022.findings-emnlp.403

Tang, Chen, Hongbo Zhang, Tyler Loakman, Chenghua Lin, and Frank Guerin. 2023. Enhancing dialogue generation via dynamic graph knowledge aggregation. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 4604–4616. https://doi.org/10.18653/v1/2023.acl-long.253

Tian, Yufei, Anjali Narayan-Chen, Shereen Oraby, Alessandra Cervone, Gunnar Sigurdsson, Chenyang Tao, Wenbo Zhao, Tagyoung Chung, Jing Huang, and Nanyun Peng. 2023. Unsupervised melody-to-lyrics generation. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 9235–9254.

https://doi.org/10.18653/v1/2023.acl-long.513

Tian, Yufei, Divyanshu Sheth, and Nanyun Peng. 2022. A unified framework for pun generation with humor principles. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 3253–3261. https://doi.org/10.18653/v1/2022.findings-emnlp.237

Touvron, Hugo, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, et al. 2023. Llama 2: Open foundation and fine-tuned chat models.

Valitutti, Alessandro, Hannu Toivonen, Antoine Doucet, and Jukka M. Toivanen. 2013. "Let everything turn well in your wife": Generation of adult humor using lexical constraints. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 243–248.

Van de Cruys, Tim. 2020. Automatic poetry generation from prosaic text. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2471–2480. https://doi.org/10.18653/v1/2020.acl-main.223

Wang, Shun, Yucheng Li, Chenghua Lin, Loic Barrault, and Frank Guerin. 2023. Metaphor detection with effective context denoising. In *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics*, pages 1404–1409. https://doi.org/10.18653/v1/2023.eacl-main.102

Wang, Shun, Ge Zhang, Han Wu, Tyler Loakman, Wenhao Huang, and Chenghua Lin. 2024. MMTE: Corpus and metrics for evaluating machine translation quality of metaphorical language. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 11343–11358. https://doi.org/10.18653/v1/2024.emnlp-main.634

Whitehouse, Chenxi, Monojit Choudhury, and Alham Aji. 2023. LLM-powered data augmentation for enhanced cross-lingual performance. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 671–686. https://doi.org/10.18653/v1/2023.emnlp-main.44

Wilshire, Carolyn E. 1999. The "tongue twister" paradigm as a technique for studying phonological encoding. *Language and Speech*, 42(1):57–82. https://doi.org/10.1177/00238309990420010301

Wöckener, Jörg, Thomas Haider, Tristan Miller, The-Khang Nguyen, Thanh Tung Linh Nguyen, Minh Vu Pham, Jonas Belouadi, and Steffen Eger. 2021. End-to-end style-conditioned poetry generation: What does it take to learn from examples alone? In *Proceedings of the 5th Joint SIGHUM Workshop on Computational Linguistics for Cultural Heritage, Social Sciences, Humanities and Literature*, pages 57–66. https://doi.org/10.18653/v1/2021.latechclfl-1.7

Wong, Min Ney, Yanky Chan, Manwa L. Ng, and Frank F. Zhu. 2019. Effects of transcranial direct current stimulation over the Broca's area on tongue twister production. *International Journal of Speech-Language Pathology*, 21(2):182–188. https://doi.org/10.1080/17549507.2017.1417480, PubMed: 29642741

Wright, Ernest Vincent. 2016. *Gadsby: A story of over 50,000 words without using the letter "E"*. Digital Ninjas Media, Inc.

Xue, Lanqing, Kaitao Song, Duocai Wu, Xu Tan, Nevin L. Zhang, Tao Qin, Wei-Qiang Zhang, and Tie-Yan Liu. 2021. DeepRapper: Neural rap generation with rhyme and rhythm modeling. 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)*, pages 69–81. https://doi.org/10.18653/v1/2021.acl-long.6

Xue, Linting, Aditya Barua, Noah Constant, Rami Al-Rfou, Sharan Narang, Mihir Kale, Adam Roberts, and Colin Raffel. 2022. ByT5: Towards a token-free future with pre-trained byte-to-byte models. *Transactions of the Association for Computational Linguistics*, 10:291–306. https://doi.org/10.1162/tacl_a_00461

Yang, Aiyuan, Bin Xiao, Bingning Wang, Borong Zhang, Ce Bian, Chao Yin, Chenxu Lv, Da Pan, Dian Wang, Dong Yan, et al. 2023. Baichuan 2: Open large-scale language models. *arXiv preprint arXiv:2309.10305*.

Yang, Bohao, Chen Tang, and Chenghua Lin. 2024. Improving medical dialogue generation with abstract meaning representations. In *ICASSP 2024 - 2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 11826–11830. https://doi.org/10.1109/ICASSP48485.2024.10447688

Yang, Bohao, Chen Tang, Kun Zhao, Chenghao Xiao, and Chenghua Lin. 2024. Effective distillation of table-based reasoning ability from LLMs. In *Proceedings of the 2024 Joint International*

Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024), pages 5538–5550.

Yao, Shunyu, Howard Chen, Austin W. Hanjie, Runzhe Yang, and Karthik Narasimhan. 2023. COLLIE: Systematic construction of constrained text generation tasks. *arXiv preprint arXiv:2307.08689*.

Yu, Dingyao, Kaitao Song, Peiling Lu, Tianyu He, Xu Tan, Wei Ye, Shikun Zhang, and Jiang Bian. 2023. MusicAgent: An AI agent for music understanding and generation with large language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 246–255. `https://doi.org/10.18653/v1/2023.emnlp-demo.21`

Yuan, Ruibin, Hanfeng Lin, Yi Wang, Zeyue Tian, Shangda Wu, Tianhao Shen, Ge Zhang, Yuhang Wu, Cong Liu, Ziya Zhou, et al. 2024. ChatMusician: Understanding and generating music intrinsically with LLM. In *Findings of the Association for Computational Linguistics*, pages 6252–6271. `https://doi.org/10.18653/v1/2024.findings-acl.373`

Zhang, Le, Rongsheng Zhang, Xiaoxi Mao, and Yongzhu Chang. 2022. QiuNiu: A Chinese lyrics generation system with passage-level input. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 76–82. `https://doi.org/10.18653/v1/2022.acl-demo.7`

Zhang, Tianyi, Varsha Kishore, Felix Wu, Kilian Q. Weinberger, and Yoav Artzi. 2020a. BERTScore: Evaluating text generation with BERT. In *International Conference on Learning Representations*.

Zhang, Ying, Hidetaka Kamigaito, Tatsuya Aoki, Hiroya Takamura, and Manabu Okumura. 2021. Generic mechanism for reducing repetitions in encoder-decoder models. In *Proceedings of the International Conference on Recent Advances in Natural Language Processing (RANLP 2021)*, pages 1606–1615. `https://doi.org/10.26615/978-954-452-072-4_180`

Zhang, Yizhe, Siqi Sun, Michel Galley, Yen-Chun Chen, Chris Brockett, Xiang Gao, Jianfeng Gao, Jingjing Liu, and Bill Dolan. 2020b. DIALOGPT : Large-scale generative pre-training for conversational response generation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 270–278. `https://doi.org/10.18653/v1/2020.acl-demos.30`

Zhuo, Le, Ruibin Yuan, Jiahao Pan, Yinghao Ma, Yizhi Li, Ge Zhang, Si Liu, Roger Dannenberg, Jie Fu, Chenghua Lin, Emmanouil Benentos, Wang Xue, and Yike Guo. 2023. LyricWhiz: Robust multilingual lyrics transcription by whispering to ChatGPT. In *Proceedings of International Society for Music Information Retrieval Conference (ISMIR)*. arXiv:2306.17103.