Thesis Defence

# A Suite of Low-Data Generation Settings Necessitating Interventions on End-to-End NLG Models

Varun Prashant Gangal

July 24, 2022

Language Technologies Institute
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15123

**Thesis Committee:**

| | |
|---|---|
| Eduard Hovy (Chair) | Carnegie Mellon University |
| Alan Black | Carnegie Mellon University |
| David Mortensen | Carnegie Mellon University |
| Sebastian Gehrmann | Google Research |
| Joel Tetreault | DataMinR |

*Submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy.*



**Keywords:** Natural Language Generation, Surface Realization, Style, Commonsense, Pragmatics, Knowledge

## Abstract

Natural Language Generation (NLG) is the field of study that aims to endow agents with the ability to generate language to satisfy any stated communicative goal (CG). Today, NLG systems that converse (e.g., Meena) and co-author (e.g., Gmail SmartCompose) with humans aren't just deployable, but a familiar part of net-equipped societies. Models underlying today's systems e.g., T5 [139], based on neural architectures like Transformers [175], are "end-to-end" in terms of their structure and overall learning process.

Notwithstanding these rapid strides, emerging work points to concerns about aspects of NLG model outputs such as, *inter alia*, commonsense plausibility [99], local coherence [124], and global coherence [88] that arise under their respective generation settings. In this thesis, we identify and characterize two distinct classes of data-deficient generation settings that present challenges for learning suitable end-to-end models when applied sans any setting-specific changes. We present six concrete instances of these classes, four from the first and two from the second:

1. **Constrained Creative Settings:** In these settings, the CG specifies an unusual, esoteric set of constraints for the outputs to satisfy, e.g., the constraint that the output should be "hard to say" i.e., phonetically difficult. Gold output examples, each of which is a commonly accepted, creatively coined artifact e.g., the tongue twister *She sells seashells on the seashore* are hard to curate, leading to low-count, small datasets (e.g., $\approx 400$ for tongue twister generation). Feasible learning inspite of such low data needs setting-driven changes to the learning process.
2. **Knowledge Deficient Settings:** In these settings, the CG requires the output to satisfy, in addition to typical requirements like fluency and input fidelity, complex aspects or properties in relation to the input such as, e.g., creating commonsense plausible combinations of input concepts for the Commongen setting [99]. Generating to satisfy these aspects needs a particularly knowledge-rich interpretation of the input. In an ideal scenario, one would expect this knowledge to be specified in the CG itself, in the form of rules, knowledge graphs or other symbolic information. However, the aspects in question are too wide-ranging in scope, making such specification impractical — Thus, the CG is in some sense "partially specified". Moreover, the training data, though not low-count, is still at a scale insufficient to acquire the knowledge tabula rasa. It is hence needed to bridge this knowledge gap by incorporating explicit sources of knowledge into the learning process such as, e.g., augmenting input via grounding in another modality for Commongen [44].

Next, we show how each setting benefits from a specific, setting-inspired *intervention* in the end-to-end nature of the NLG model architecture and learning process to design a final, improved NLG system that viably generate outputs sufficiently satisfying the CG. Finally, we sketch out a general recipe outlining how to design such interventions. The sheer diversity of linguistic form means there will always arise new, data-deficient NLG settings that involve unusual constraints or underspecified CGs. This thesis illustrates a general method for addressing such situations systematically.

# Contents

# List of Figures

# List of Tables

# Chapter 1

# Introduction

The old order changeth yielding place to new;
And God fulfills himself in many ways,
Lest one good custom should corrupt the world.

---

Lord Alfred Tennyson

The old that is strong does not wither,
Deep roots are not reached by the frost.

---

JRR Tolkien

## 1.1 Motivation

Natural Language Generation (NLG) is a field of study which aims to endow machines with the ability to generate human language to satisfy a stated communicative goal (CG). The CG can encompass given input information, constraints on the output, and a multitude of other components and specifications. NLG is a subfield of Natural Language Processing (NLP), and is often seen as a dual to Natural Language Understanding (NLU). Solving NLG is a key prerequisite to realizing the overall goal of building an “AI-Complete" machine.

Notwithstanding the steady progress made in recent years by models instantiating the end-to-end trainable neural NLG pipeline (that we shall also refer to as E2EN2PP), emerging work points to concerns about aspects of their outputs, both in isolation as well as in the way they relate to

their input and other CG components. Such aspects include, *inter alia*, commonsense plausibility [99], local coherence [124], and global coherence [88]. This growing body of work underscores the importance of identifying families of settings and situations where state-of-the-art models instantiating the E2EN2PP cannot be deployed in direct fashion with satisfactory results.

Motivated by this, through this thesis, we aim to identify and characterize two such distinct classes of data-deficient generation settings that present challenges for learning suitable end-to-end models.

1. **Constrained Creative Settings:** In these settings, the CG specifies an unusual, esoteric set of constraints for the outputs to satisfy, e.g., the constraint that the output should be “hard to say" i.e., form a difficult sequence of phonemes. Gold output examples, each of which is a commonly accepted, creatively coined artifact e.g., the tongue twister *She sells seashells on the seashore* are hard to curate, leading to low-count, small datasets (e.g., $\approx 400$ for tongue twister generation). Training an E2EN2PP to produce such output inspite of such low data needs changes to the learning process.
2. **Knowledge Deficient Settings:** In these settings, the CG requires the output to satisfy, in addition to typical requirements like fluency and input fidelity, complex aspects or properties in relation to the input such as, e.g., creating commonsense plausible combinations of input concepts. Generating to satisfy these aspects needs a particularly knowledge-rich interpretation of the input. In an ideal scenario, one would expect this knowledge to be specified in the CG itself, in the form of rules, knowledge graphs or other symbolic information. However, the aspects in question are too wide-ranging in scope, making such specification impractical — Thus, the CG is in some sense “partially specified". Moreover, the training data, though not low-count, is still at a scale insufficient to acquire the knowledge tabula rasa. It is hence needed to bridge this knowledge gap by incorporating explicit sources of knowledge into the learning process.

First, we provide a set of definitions for the main terms employed in the thesis.

Next, in total, we present six concrete instances of these classes, four from the first and two from the second: We show how each setting benefits from a specific, setting-inspired *intervention* in the end-to-end nature of the NLG model architecture and learning process to design a final, improved NLG system that viably generate outputs sufficiently satisfying the CG.

Finally, we sketch out a general recipe outlining how to design such interventions. The sheer diversity of linguistic form means there will always arise new, data-deficient NLG settings that involve unusual constraints or underspecified CGs. In conclusion, this thesis illustrates a general

method for addressing such situations systematically.

## 1.2 Terms & Definitions

### 1.2.1 Preliminaries & Guiding Principles (Generic Definitions)

1. **Grammar, Grammaticality & Fluency:**

   A *grammar* is a set of rules of the form $\Sigma^* \rightarrow \Sigma^*$, defined over an alphabet of symbols $\Sigma = NT \cup T$, where $NT$ are the non-terminal symbols and $T$ are the terminal symbols.

   We call a piece of text *grammatical* or say that it possesses *grammaticality* if it can be generated by the grammar of English, or the language under question. Note, however, that it is an almost impossible task to write a grammar for an entire existing, sui-generis language which handles all sentences/phenomena seen in that language, though even the earliest grammarians like Panini [149] have made attempts at this. As a result, when we say *grammatical* what we mean is that the piece of text would be considered *acceptable* by most native speakers of the language when they are asked so, and is hence also sometimes called *acceptability*. A related but slightly different notion is that of fluency — note that the slight difference here arises from the well-studied competence (acceptability) vs performance (fluency) distinction [118] in linguistics — i.e text is fluent is if it sounds like a natural text you would hear from a native speaker of the language — such texts of course would be largely grammatical , but it would also exclude grammatical sentences which are meaningless i.e., they are so implausible that it is hard to assign them a meaning, even an abstract or imaginative one e.g., Chomsky's famous example *Colorless green ideas sleep furiously.*

2. **Language Model:**

   A language model (LM) defines a probability distribution $P(s)$ over all possible word (or subword/character, depending on modelling choice and task etc.,) sequences $s \in S$, where $S$ is the Kleene closure of the vocabulary $V$.

   Many LM architectures are factored in left-to-right fashion $P(s) = \Pi_{i=2}^{i=|s|} P_{next}(w_i|s_1^{i-1})$, where $P_{next}$ is the next-word distribution and $s_i^{i-1}$ is the subsequence of the first $(i-1)$ elements of $s$. Note that there also exist other formulations, e.g., whole sentence language models [150].

3. **Transducer**

   A transducer is a model $f_\theta() : V_{in}^* \times X \rightarrow V_{out}^*$ which can accept an input sequence string $s_{inp} \in V_{in}^*$ from an input vocabulary $V_{in}$, where $*$ is the Kleene closure and $\theta$ are the transducer's parameters, along with other potential inputs/parts of the communicative goal (themselves

symbolic or continuous) $x_{in} \in X$, and output a sequence $s_{out} \in V^*_{out}$. The transduction function can be written as $x_{out} = f_\theta(s_{in}, x_{in})$

Any NLG model basically functions as a transducer at inference time/test-time (taking in the communicative goal and returning the output sequence), though the internal representations, learning process and architecture can vary significantly.

When $X = \phi$ and and $V_{in} \subset V_{out}$, it becomes possible to use any left-to-right factored language model architecture as a transducer. This is since one can now feed in $s_{in}$ as the first few tokens of a segment to function as a "prompt" to the language model (this part is "teacher forced" running of the model, i.e., since the sequence is predetermined, the language model is only fed the sequence under question) and then predict out $s_{out}$ using the language model using some decoding method (see §2 for more). Note that the typical traditional view in NLG requires X to always be non-empty since the host system invoking the NLG model always has atleast something computed which it needs to convey to the model.

Transducer models which have two roughly separatable modules can be called Seq2Seq or encoder-decoder models. The first module, or the *encoder* is for representing $s_{in}$ in some symbolic or continuous intermediate form $h$ (Note that $h$ could even be a sequence or set of things, e.g., a set of vectors). The second module uses $h$ to then generate $s_{out}$ — this module is known as the *decoder*. The term Seq2Seq is also often used in a wider sense for any neural transducer and not just the particular form above. Seq2Seq models where the decoder uses some internal form of attention mechanism [5] are also described as *attentional*.

4. **Infilling**

   Intuitively, infilling refers to the process of using a learnt model to perform "fill in the blanks" i.e., predicting a masked out token (usually using a special character e.g., *[MASK]*) given its surrounding context. Depending on the model architecture and training, this might involve the entire left and right contexts or subsets of them (e.g., only the left context for left-to-right language models).

5. **Systemic Functional Linguistics**

   Systemic Functional Linguistics (SFL) was a theory devised by the linguist M.A.K. Halliday in the 1970s [58] SFL categorizes subgoals or subparts of the communicative goal into three metafunctional categories:

   (a) **Ideational Goals:** These subgoals pertain to the author's state of mind; their knowledge, memory and experience about the various states of the world (factual, physical etc.,) inter alia .

   (b) **Interpersonal Goals:** These subgoals pertain to the relationship between the speaker and

the listener/addressee. It also subsumes subgoals pertaining to the medium of transmission, or the individual physical / emotional states of the addressee

(c) **Textual Goals:** These subgoals pertain to choices in terms of the order of presentation of information in the text, the subset of textual surface forms employed, and the internal structure and packaging of the text in terms of its constituent sentences, phrases, words and other elements.

6. **Rhetorical Goals**

The non-textual subgoals of the wider communicative goal, namely those which can be categorized under the *Ideational* and *Interpersonal* metafunctions are also sometimes referred to as *Rhetorical Goals*.

### 1.2.2 Defining a NLG System

Having laid out preliminaries, guiding principles, and other generic definitions, we shall now define a NLG system and associated concepts.

#### Communicative Goal (CG)

The overall goal which the output of the NLG system must satisfy in order for the process of generation, and consequently the model, to be deemed successful. This also includes all the information which the NLG model needs to modify, process and condition on while generating its output.

It is common to characterize and address certain parts (or subgoals) of the CG as controls, input and style(s) etc., though the choice of these parts is highly subjective in nature — for example, for a movie review, the sentiment is considered part of the “input" when doing formality transfer, but is considered a “control/style" when doing sentiment transfer.

#### Subtasks

The subtasks of natural language generation are a conceptual decomposition of the activities to be performed to generate a text, given a CG. They may also be thought of as subgoals to be accomplished before the overall CG has been achieved.

1. **Content Selection:**

Before generating the sentences and words, it is necessary to decide “What all to say?" out of all the potential things which suitably fit the communicative goal. The set of these choices is called content selection.

2. **Content Ordering:**

   Having decided what to say, it is necessary to decide “In what order?" the selected pieces of information from content selection would be presented in. The process of chooosing this is content ordering. Collectively with content selection, the two are also referred to as Macroplanning or Sentence Planning. This can be heavily dependent on the rhetorical goals (roughly speaking, extra-textual goals; see §6 for a complete description) e.g., For a Twitter thread, it might be required to place more retweetable and topically high-coverage content earlier on.

3. **Sentence Aggregation:**

   This subtask pertains to the breaking up and packaging of the content to present into sentences, according to the broad order decided in Content Ordering.

4. **Lexicalization:**

   This subtask refers to the choice of which word forms to broadly use in each sentence. Note that some subdecisions maybe left unspecified for the latter stages, especially Surface Realization.

5. **Referring Expression Generation:**

   Referring Expression Generation, a.k.a. *Refex Generation*, is the choice of expressions, or refexes to point to various entities, events or other item types while mentioning them at each point in the generation. (this can include pieces of the generated output itself i.e., discourse segments e.g., as in “In our earlier argument, . . . ")

6. **Surface Realization:**

   Also referred to simply as *Realization*, this refers to the final, explicit generation of the output text, resolving all the partially specified elements from earlier stages, as well as filling in remaining gaps based on syntactic, co-occurrence based, prosodic and other considerations.

Though there is a natural ordering and sequence to the subtasks based on their typical mutual dependency, and that is the order in which we shall present them, they need not always be performed in that order, though the Classical NLG Pipeline which we shall describe in 1.3.4 makes a best attempt to do so. For instance, for many a CG, Referring Expression Generation might be entirely independent of Lexicalization. For some others, it may be very closely tied to syntax (e.g., in pro-drop languages like Spanish) (and hence would need to be revised) during Surface Realization.

**End-to-End Neural NLG Pseudo Pipeline**

End-to-End Neural NLG Pseudo Pipeline or E2EN2PP refers to the canonical neural architecture for NLG based on the Seq2Seq paradigm, where one or more encoders first encode the CG. Next, the encoder representations are aggregated or functionally transformed in various ways. Finally, a decoder network uses any of the encoder representations to compute the probability/loss functions based additionally off the gold output (at training time) or to generate the output text at test-time using some inference/search procedure or sampling method.

Note that the Seq2Seq paradigm in general, and our characterization of it here in particular, though general enough to include many paradigms of neural architectures, do not cover all of them exhaustively — particular exceptions being VAEs, GANs, Energy-Based Models etc. We leave performing a similar study on these models with a generalization of our framework, as we do in this thesis, as a point for future work.

**Concept-To-Text-Generation Tasks**

These are tasks where the CG requires generating a pertinent output of one or a few sentences, given a largely unstructured or semi-structured collection of “concepts" as an input. We will also refer to the input in such tasks as *concept set* or *input concept set*. Commongen [99] (where the task is to generate a single sentence describing a situation involving all the given concepts) and WebNLG [52] (where the task is to describe a sequence of SVO triples) are two prominent examples of this family of tasks.

## 1.3 Contributions & Structure

Having defined the supporting terms and concepts, we now proceed to describe in detail the two classes of data-deficient NLG settings we motivated in §1.1 and summarize a total of six instances across these settings. Each instance summary also includes a brief description of the associated **Intervention** to the E2EN2PP.

The rest of this thesis is split into two parts — Part I (Constrained Creative Settings, see §1.3.1) and Part II (Knowledge Deficient Settings, see §1.3.2). Each part identifies and characterizes a class of data-deficient NLG settings which require interventions to the end-to-end nature of the E2ENLPP. The parts are further subdivided into chapters, each of which present a new generation setting exemplifying the respective part. We shall present four chapters in Part I . This shall be followed by two chapters in Part II.

### 1.3.1 Part I: Constrained Creative Settings

#### Preface

A large fraction of natural language consists of a stream of fluent, plausible-sounding and topically consistent text intended to communicate the author's underlying message while adhering to certain basic constraints such as media constraints, interpersonal norms and the Gricean maxims.

However, a smaller, though significant, fraction of natural language also abounds with creative devices of expression e.g., metaphors, idioms, tongue twisters, portmanteaus and personification. Through use of these devices, the author can more actively engage the reader and fulfill other subgoals beyond mere communication such as being memorable, persuasive etc.

A truly AI-complete NLG system with abilities close to a human speaker must also be capable of performing in settings where it is required to generate one of these specific devices e.g., generating a tongue twister given a short textual prompt or a set of keywords, as we shall discuss in Chapter 4.

A first defining property of such creative generation settings is the atypical, unusual or esoteric set of output constraints specified by the CG. e.g., for tongue twisters the constraint that the output should be "hard to say" i.e., constitute a phonetically difficult sequence of phonemes, which we shall further study in Chapter 4. The CG constraints in this class of settings also often entails thinking about a deeper layer of representation beyond the words (and letters) and sentences themselves, such as for the above case, the *phonetics*.

A second defining property of such creative generation settings is the *data availability*, specifically, the small size of training data available. This property naturally arises from the low count of widely known creative artifacts in that phenomena which were ever creatively coined and came to garner wide acceptance. For some settings, even the few examples available are deficient and require imputation, such as e.g., in the setting of generating a personification-endowed sentence from a source sentence which originally lacks personification, which we shall study in Chapter 3. Here, at training time, we only have access to individual sentences which already exhibit personification e.g., *My alarm clock yells at me every morning* — we lack access to paired examples of de-personified sentences and their personified counterparts.

A final defining property of these settings is the recipe through which one can overcome the paucity of training data and construct a viable NLG model whose outputs sufficiently satisfy the CG. Specifically, this requires modifying the typical E2EN2PP pipeline by means of a *intervention* incorporating motivation from a "*creative story*" underlining the setting i.e an underlying

theory/schematic model that characterize how a typical human speaker might have performed the creative phenomenon. Consider, for instance, the setting of generating a portmanteau given a pair of root words, which we shall further study in Chapter 4. A portmanteau is a blend of the two root words that sounds lexically and phonetically like a typical English word, while at the same time being a useful shorthand, i.e., reminiscent of the root words. The *creative story* here would involve the human speaker utilizing two internal congitive models in unison – A first model acquired from the English vocabulary of which candidate character sequences are word-like, and a second model which checks whether the root words are “guessable" given a character sequence. Inspired by this, the *intervention* we perform here involves a Bayesian-style **refactoring** from the typical, forward factored model P(y|x) to a Noisy Channel Style Model P(x|y)P(y). We can now enforce a “sounding word-like" bias by using English vocabulary word lists to **pretrain prior P(y)**, much akin to the speaker's first model from the creative story. Furthermore, the P(x|y) more directly captures the aspect of the portmanteau $y$ being reminiscent of the root words $x$, akin to the speaker's second internal model.

To summarize, each chapter/setting in this class of Constrained Creative Settings can be characterized as being its members in terms of three defining properties.

1. **CG Definition/Constraints**
2. **Data Availability**
3. **Intervention**

### Chapter Outline

This part of our thesis contains four chapters.

- **Chapter 2: Portmanteau Generation**
- **Chapter 3: Personification Generation**
- **Chapter 4: Tongue Twister Generation**
- **Chapter 5: Style Transfer To Shakespearize Modern English**

An exhaustive, summarizing recap of all the ensuing chapters descriptions can be found in Table 1.1

First, in **Chapter 2**, we explore the setting of portmanteau generation. Portmanteaus are a creative word formation phenomenon where two words blend to form a new word, with a meaning derived from but distinct to their original meanings e.g., *wiki* + *etiquette* $\rightarrow$ *wikiquette*, *fashion* + *fascism* $\rightarrow$ *fashism*.

| Chapter/Setting | CG Definition/Constraints | Data Availability | Intervention |
|---|---|---|---|
| Ch. 2<br>Portmanteau<br>Generation | Given two root words, generate<br>a single word i.e. a blend that is<br>i) Lexically & Phonetically Word-Like<br>ii) Reminiscent of/faithful to root words<br>**Deeper Layer:** Phonetics | 400, Input+Target | i) **Refactoring** from a forward factored model P(y\|x)<br>to Noisy Channel Style Model P(x\|y)P(y)<br>ii) Enforcing a "sounding word-like" bias<br>by using word lists to **pretrain prior P(y)**, implicitly<br>inducing English-like phonetic preferences |
| Ch. 3<br>Personification<br>Generation | Given a sentence lacking prior<br>personification, introduce personification<br>by assigning an inanimate entity<br>animacy-requiring attributes/roles<br>as per the underlying dependency<br>structure<br>**Deeper Layer:** Dependency Structure | 350, Target-only | i) Devising a "de-personification" pipeline<br>**to construct noisy source-side inputs** $x_{noisy}$ using<br>dependency parsing, commonsense knowledge bases<br>and pre-trained BART infilling followed by a<br>3-component reranking heuristic to enforce loss of<br>animacy, fluency whilst preserving meaning<br>ii) Using **the constructed, pseudo-parallel pairs** $\{x_{noisy}, y\}$<br>to finetune a well-pretrained transduction models,<br>e.g., BART, T5 etc |
| Ch. 4<br>Tongue Twister<br>Generation | Generate a sequence of graphemes<br>i) Forms a fluent, meaningful<br>sentence<br>ii) Is also articulatorily difficult i.e.,<br>forms a sequence of phonemes that is<br>"hard to say"<br>**Deeper Layer:** Phonetics | 644, Input+Target | i) Devising **a phoneme-aware finetuning mechanism**<br>to both leverage a) Ability of strongly pretrained<br>grapheme-only models to fluently complete prompts<br>b) Instilling into them the notion of phonetic hardness<br>by **heterogenously training** to generate either of<br>phoneme/grapheme completions from prompts in either<br>of phoneme/grapheme forms (G2P,P2G,G2P,P2P)<br>ii) At inference time, use grapheme-to-grapheme (G2G)<br>mode to infer from learnt model. The other<br>3 finetuning modes merely served as **scaffolding** |
| Ch. 5<br>Style Transfer<br>To Shakespearize<br>Modern English | Given an English source sentence,<br>transduce it lexico-syntactically to sound like<br>Modern English while preserving meaning<br>**Deeper Layer:** Lexical/Phrasal Mappings | 10000, Input+Target | Incorporating pairwise lexical knowledge<br>through incorporating **pairwise constraints** over<br>a **shared, pretrained embedding space**<br>using **the Retrofitting procedure** [39] |

Table 1.1: An Exhaustive Tabular Recap outlining all the settings we study under the class of **Constrained Creative Settings** (Part I) — Chapters 2, 3, 4 and 5

- **CG Definition/Constraints:** Given two root words, form a single word neologism i.e. a blend that is
  1. Lexically and phonetically “word-like"
  2. Is an effective shorthand i.e., is reminiscent of and faithful to root words
- **Data Availability:** We have access to $\approx$ 400 examples of portmanteaus along with their root words.
- **Intervention:** We first form an intuition for how intervene based on an underlying “creative story". The *creative story* here would involve the human speaker utilizing two internal congitive models in unison – A first model acquired from the English vocabulary of which candidate character sequences are word-like, and a second model which checks whether the root words are “guessable" given a character sequence.
  1. Bayesian refactoring from a forward factored model P(y|x) to Noisy Channel Style model P(x|y)P(y)
  2. Enforcing the “sounding word-like" bias from the CG constraint as well as the speaker’s first cognitive model from the creative story by pretraining the prior P(y) on all character sequences corresponding to English vocabulary word types.

  Specifically, the intervention to the E2E NLG pipeline required here, is fleshed out in Figure 1.7.

This work on its completion was accepted for publication as a short paper at EMNLP 2017. The respective publication is [48].

Next, in **Chapter 3**, we study the setting of personification generation. A personification is a figure of speech that endows inanimate entities with properties and actions typically seen as requiring animacy.

- **CG Definition/Constraints:** Given a literal source sentence lacking prior personification, introduce personification by assigning an inanimate entity animacy-requiring attributes/roles. This CG constraint points to the importance of the deeper layer of underlying dependency structure in the target-side, personified sentences.
- **Data Availability:** As training data, we only have access to $\approx$ 350 examples of target-side, personified sentences. Note that this data, besides being low count, is also incomplete in terms of the typical source-target parallel data used to train end-to-end models — we lack access to source-side, depersonified input sentences.
- **Intervention:** We first hypothesize a creative story based on a TOPIC-ATTRIBUTE

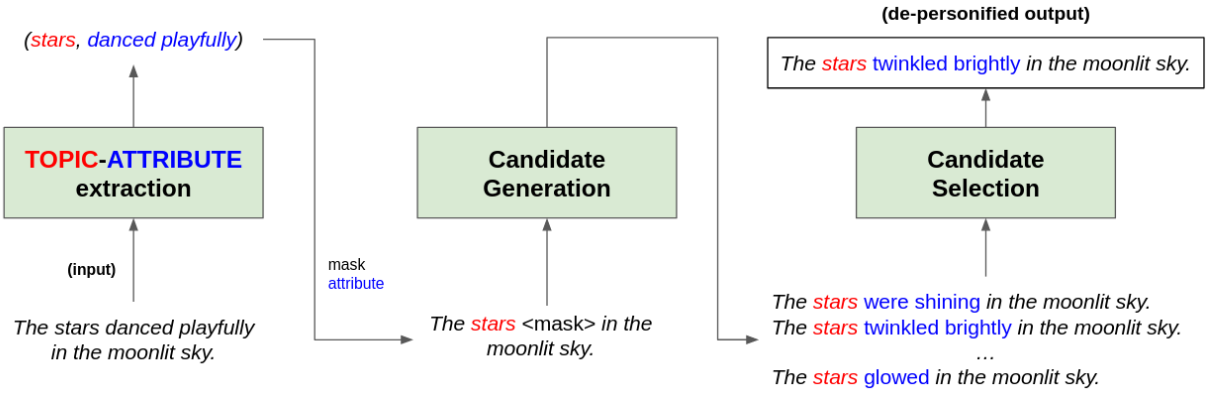



Figure 1.1: Overview of the de-personification pipeline.

relationship between the TOPIC, which is an inanimate entity and its animacy-requiring ATTRIBUTE, which is a dependent based on the dependency structure. This TOPIC-ATTRIBUTE structure is illustrated through examples in Figure 1.3.

Based on this creative story, we devise a two-step intervention to the E2EN2PP corpus curation and consequently, the overall training process.

1. Devising a “de-personification" pipeline to construct noisy source-side inputs $x_{noisy}$ which replace the animacy-requiring portions of the sentence with equivalent animacy-agnostic ones. using off-the-shelf dependency parsing, commonsense knowledge bases and pre-trained BART infilling followed by a 3-component candidate heuristic to enforce loss of ATTRIBUTE animacy, whilst preserving fluency and meaning
2. Using the constructed, pseudo-parallel pairs to finetune a well-pretrained transduction model checkpoint, e.g., BART or T5.

Specifically, the intervention to the E2EN2PP corpus curation and overall training process required here, is fleshed out through Figures 1.1 and 1.2 respectively. The de-personification pipeline shown in Figure 1.1 is in turn based on the underlying TOPIC-ATTRIBUTE schematic model/“creative story" depicted through examples in Figure 1.3.

This chapter on its completion was submitted as a long paper to COLING 2022 and is currently under review.

Moving further, in **Chapter 4**, we study the setting of tongue twister generation from short prompts/keywords. Tongue twisters are meaningful sentences that are difficult to pronounce e.g., *She sells seashells on the seashore*. The process of automatically generating tongue twisters is challenging since the generated utterance must satisfy two constraints at once: phonetic difficulty

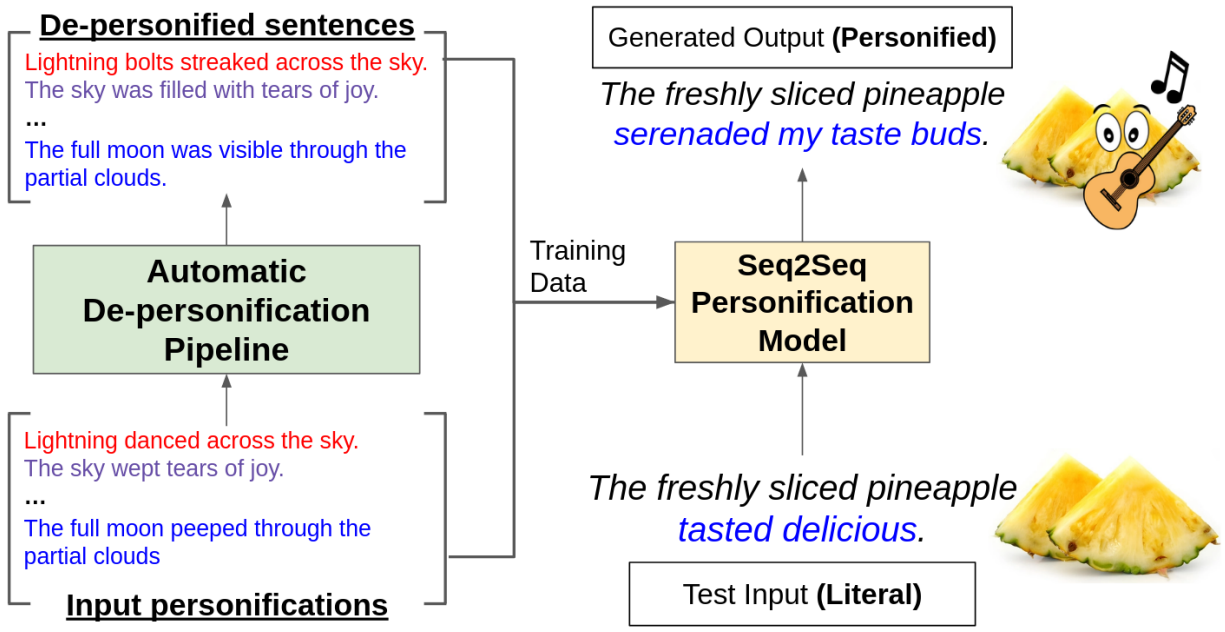



Figure 1.2: Overall model pipeline after intervention for the setting in Chapter 3 . The left part of the diagram shows the atypical corpus curation process, which represents an intervention to the E2EN2PP. Instead of using naturally available parallel data, which is missing in this setting, an Automatic De-personification Pipeline, further described in Figure 1.1 is used to construct noisy inputs (de-personified sentences). The right part of the diagram shows the training and generation process.

| ATTRIBUTE Type | Example |
|---|---|
| Noun | The planet earth is our mother. |
| Verb | My alarm clock yells at me to get out of bed every morning. |
| Adjective | Justice is blind and, at times, deaf. |

Figure 1.3: Examples of different variations of personification with their respective ATTRIBUTES and TOPICS, as proposed by our TOPIC-ATTRIBUTE formulation (TOPICS in red and ATTRIBUTES in blue).

and semantic meaning. Furthermore, phonetic difficulty is itself hard to characterize and is expressed in natural tongue twisters through a heterogeneous mix of several phenomena such as alliteration and homophony.

- **CG Definition/Constraints:** Given a short textual or keyword prompt, generate a completing sequence of graphemes such that it
  1. Forms a fluent, meaningful sentence
  2. Is also articulatorily difficult i.e., forms a sequence of phonemes that is “hard to say".
- **Data Availability:** The coining of a novel, unique tongue twister that spreads sufficiently to become normative and well-recognized is rare, hence making them a long-tailed linguistic phenomena [117]. As training data, we only have access to $\approx 644$ examples of tongue twisters including their accompanying prompts.
- **Intervention:** We first hypothesize a creative story. Consider the idealized scenario where we have a *"mouth model"* that a) maps different regions of the mouth, palate, and the larynx to the dictionary of fundamental sounds, i.e. phones being produced, and b) based on this grounding, can quantify the hardness of producing one sound after another, i.e. induces a distance measure between any phone pair. Assuming access to this idealized model, one could deconstruct the process of generating a tongue twister as sampling a sequence of preferably difficult, i.e. distant phone-phone transitions starting with an initial sequence of one or more phones. However, there are impediments that make realizing such an idealized model considerably intractable. Firstly, the dictionary of fundamental sounds at the granularity we use in practice, i.e. at the level of phonemes, does not neatly map to particular points of the palate [92]. Secondly, a tongue twister as per its definition is not merely a difficult to pronounce sequence of phonemes — but also one that maps

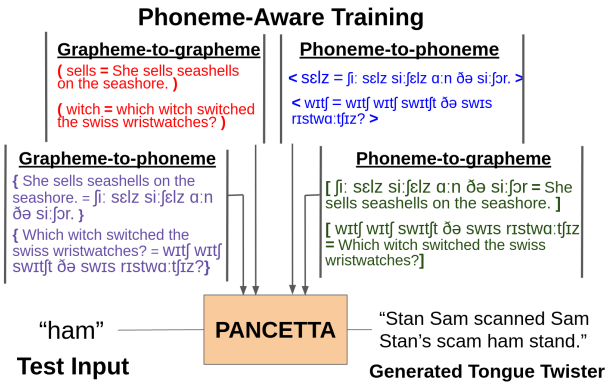



Figure 1.4: Overview of the phoneme-aware, scaffolded training mechanism in the post-intervention “PANCETTA" model from Chapter 4. Note how the typical Grapheme-to-grapheme mode is the one used for actual inference, with the other three modes serving as training-time only “scaffolding", making the model phoneme-aware and biased towards phonetic hardness.

to a meaningful and fluent sequence of words. How one can maintain this property in conjunction with the process of sampling difficult transitions from the mouth model’s space is not immediately clear.

However, one answer, could lie with using strong pretrained models such as GPT-2, which are already primed to generate fluent and meaningful completions in grapheme space. If only we had a way to make them phonetically aware, they could be finetuned towards phonetic hardness. Based on this intuition, we devise a two-step intervention to the typical E2EN2PP training process.

1. At training time, instead of finetuning GPT-2 in grapheme-to-grapheme (G2G) mode alone, as would be typical, we heterogenously finetune it to generate either of phoneme/-grapheme completions from prompts in either of phoneme/grapheme form.
2. At inference time, use grapheme-to-grapheme (G2G) mode to infer from learnt model. The other 3 finetuning modes (G2P, P2G,G2G) merely served as a training-time, “scaffolding" mechanism.

Specifically, the intervention to the E2EN2PP training process required here, is fleshed out through Figure 1.4.

This work on its completion was submitted as a long paper to EMNLP 2022 and is currently under review.

Finally, concluding Part I, in **Chapter 5**, we address the aspect of *controlling diachronic register*, which is expressed primarily through *lexico-phrasal means*. Specifically, we explore the task of transferring the style of a given sentence authored in contemporary English, such as

e.g. *I am in a rush*, to the style of William Shakespeare, who wrote in the Early Modern English prevalent in Elizabethan times, such as e.g., *I stand on sudden haste*.

- **CG Definition/Constraints:** Given an English source sentence, transduce it lexico-syntactically to sound like Early Modern English while preserving the source's original meaning.
- **Data Availability:** We have access to $\approx$ 10000 parallel source-target pairs. This is an order of magnitude lower than the typical counts of parallel data used to train strong machine translation models. Additionally, we have access to a noisy, sparse lexicon of $\approx$ 1000 source $\rightarrow$ target lexical correspondences, e.g., *thou* $\rightarrow$ you.
- **Intervention:**
  1. Tying together the source and target side embedding spaces after forming a unified vocabulary. The shared embedding space is now pretrained on a combination of the source and target-side sentences.
  2. Incorporating pairwise lexical knowledge into the E2EN2PP pipeline through incorporating pairwise constraints such as e.g., *thou* $\rightarrow$ *you* into the shared, pretrained embedding space

This work on its completion was accepted as a long paper at the EMNLP 2017 Workshop on Stylistic Variation. The respective publication is [74].

An exhaustive, summarizing recap of all the chapters from the current part can be found in Table 1.1

### 1.3.2 Part II: Knowledge Deficient Settings

**Preface**

For many NLG settings, the input is employed by the CG to merely provide the direct, basic textual context or "bounding box" within which generation takes place, with the CG in addition specifying/laying out the remaining conditions and constraints an output should satisfy, either explicitly or inductively through the choice and diversity of training examples provided. This includes common settings such as abstractive summarization, where the input document provides the overall content, while the rest of the CG explicitly specifies the constraints on the summary such as length/style (tweet vs headline?) and additional controls (such as a question in query-based summarization). This also includes highly constrained creative settings of the nature we studied in Part I.

In this part, we shall focus on a different, challenging class of NLG settings where the CG

specifications require using, interpreting and representing the input in greater complexity. In these settings, the CG requires the output to satisfy, in addition to typical requirements like fluency and input fidelity, *complex aspects* or properties in relation to the input such as, e.g., creating *commonsense plausible* combinations of input concepts for the Commongen setting [99], which we shall be studying in Chapter 6.

Generating to satisfy such *complex aspects* in relation to the output needs a particularly *knowledge-rich* interpretation of the input. In an ideal scenario, one would expect this knowledge to be specified in the CG itself, in the form of additional symbolic machinery it develops to further ground the input w.r.t the desired aspect, such as through rules, knowledge graphs or other symbolic information. Notions such as commonsense plausibility and relevance towards game pragmatics maybe too vast and wide-ranging to encode or represent by these traditional means, whether in terms of the size of the symbolset needed or the count of interactions and dependencies the symbols would have to encode. For instance, consider having to define explicitly the game pragmatics of chess. One may, with some difficulty, define the game rules e.g., what constitutes a valid move, special moves (castling and en passant), end conditions etc. However, it would be an even more intractable pursuit to define the game's conventions e.g., typical opening moves, terms used to describe pairwise piece configurations (*developed* and *blocked*) etc.

A complementary, implicit way the CG can specify some NLG settings is inductively i.e through the nature and diversity of input/output pairs provided for training. However, for the class of settings we study, the training data, though not low-count, is still at a scale insufficient to acquire the requisite extent of knowledge *tabula rasa*.

Thus, for this class of settings, the CG is in some sense "partially specified". It is hence needed to bridge this knowledge gap by explicitly incorporating another *approximate knowledge source* into the learning process by intervening in the E2EN2PP.

To summarize, each chapter/setting in this class of e settings can be characterized as being its members in terms of three defining properties.

1. **Complex CG Aspect**
2. **External Knowledge Source**
3. **Intervention**

### Chapter Outline

This part consists of two chapters

| Chapter/Setting | Complex CG Aspect | Knowledge Source/Intervention |
|---|---|---|
| Ch. 6 VisCTG: Improving Plausibility of Microplanning for Concept-To-Text Generation Through Retrieve-Caption-Generate | Commonsense Plausibility of output situation | **Knowledge Source:** Visual modality<br>**Intervention:**<br>ii) Issue input concept set as query to search engine → Generate captions from top K most relevant images → Augment input<br>iii) Input Expansion Layer between Input & Embedding Layers |
| Ch. 7 Viable Content Selection For and Refex Generation Through Pragmatic Backoff For Chess Commentary Generation | Game Pragmatics ought to be evoked while describing the move in the commentary sentence | **Knowledge Source:** Chess game library (pychess) which can enumerate pragmatically pertinent piece-piece interactions<br>**Intervention:**<br>i) **Incorporate pragmatic knowledge** through chess game library<br>ii) ***Pragmatic Interpretation Layer*** to featurize / alter inputs before passing onto End to End Pipeline (See Figure 1.8) |

Table 1.2: An Exhaustive Tabular Recap outlining all the settings we study under the class of **Knowledge Deficient Settings** (Part II)

- **Chapter 6: VisCTG: Improving Plausibility for Commongen Through Retrieve-Caption-Generate**
- **Chapter 7: Viable Content Selection and Refex Generation Through Pragmatic Backoff For Chess Commentary Generation**

First, in **Chapter 6** we study the Commongen [99] setting, where the CG is to generate a sentence constructing a commonsense plausible situation from a given set of input concepts.

We first identify several critical issues in baseline model generated outputs for this task, like poor plausibility, inadequate lexical relationships, and incomplete arguments etc. We posit that properties specific to the textual modality, such as the Gricean maxim of Quantity and the Zipfian nature of concept occurrence, could indeed have a marked negative downstream effect on the NLG model's learning for CommonGen.

We posit that using the visual modality as an **External Knowledge Source** could be advantageous. We devise an **Intervention** that augments the input concepts by drawing information from the visual modality to help dampen this negative effect. Specifically, the intervention we devise to the E2EN2PP is the addition of an Input Expansion Layer between the Input and the Embedding Layer. Before passing the input string to the Embedding Layer, the Input Expansion Layer symbolically augments it with the captions of retrieved relevant images. Figure 1.9 illustrates our intervention.

A work based on this chapter was accepted as a long paper at AAAI 2022 [43].

Finally, in **Chapter 7**, we study the NLG setting where the CG requires *generating a short, interesting natural language commentary sentence for each chess game move during gameplay*. We show how S2S models simply based on a E2EN2PP suffer from the common response problem [34] and fail to produce commentary that is even at the level of a template based baseline. We posit that this arises from the inability to acquire tabula rasa the pragmatic knowledge necessary

to understand the input game state.

We posit that using game libraries such as *pychess*, which can extract and enlist pragmatically pertinent piece attributes and inter-piece interactions, as an **External Knowledge Source** could be advantageous.

We devise an **Intervention** that includes an additional Pragmatic Interpretation Layer to discretely featurize the board states using the pychess game library, essentially backing off to pragmatic game knowledge to viably declutter the input states, thereby simplifying the understanding and overcoming the microplanning and macroplanning issues observed. The devised intervention in the E2EN2PP that needs to be done can be seen in Figure 1.8.

A work based on this chapter was accepted as a long paper at ACL 2018 [75].

An exhaustive, summarizing recap of all the chapters from the current part can be found in Table 1.2

### 1.3.3 Summary of Chapter

In summary, this chapter first started out by motivating the need to study situations where the E2EN2PP in its typical form proves insufficient. It then introduced two distinct classes of such data-deficient NLG settings, the former characterized by unusual CG constraints on its output, and the latter characterized by complex, knowledge-heavy CG constraints on the output-input relationship accompanied by an underspecified CG, both of which require interventions to the E2EN2PP to overcome the lack of data. After laying out the basic terms and definitions, it then illustrated the range of such settings by summarizing six specific instances spread across the two classes, each summary also enclosing a brief description of the required intervention to the E2EN2PP. The description of the intervention within each class, is presented following a common "recipe" or guiding thought process. The individual chapters which follow will now flesh out each instance in complete detail.

### 1.3.4 Appendix

#### Additional Terms & Definitions

Below, we lay out additional terms and definitions to further support the ones we already covered earlier in the chapter.

1. **Finite State Automatons**

   An automaton is a finite-memory (or one-step memory) accept-reject mechanism that can take

in a sequence of symbols from some symbolset $\Sigma$ as input. Internally, the automaton consists of a i) set of states ii) transition arcs between states which are traversed based on the current input symbol read iii) start and end states ; note that these are from the existing states and may themselves overlap

Symbol sequences which on being read by the automaton take it to one of its stop states are said to be accepted by the automaton. Every automaton has a corresponding CFG associated with it. These automatons are also known as *Finite State Automatons* (FSAs).

A generalization of FSA is that of pushdown automata, which are also provided with a stack of potentially unbounded length.

*Finite State Transducers* (FSTs) are FSAs which can also emit output symbols during transitions (or after reaching an input state, depending on how one may define it)

In Chapter 2, we will see FSTs being employed by one of the baseline approaches.

2. **Decoding**

   Decoding refers to the algorithm which finally uses a learnt NLG model to produce the output given an instance of the communicative goal (or “input" instance, if we assume the rest of the communicative goal to remain fixed for the “task") .

3. **Gricean Maxims**

   The four Gricean maxims of *Quality*, *Quantity*, *Manner* and *Relation* are four general guidelines relating the pragmatics of the speaker and their actual utterance; implicitly followed by most human speakers (though sometimes violated intentionally, e.g., for humour) They are sometimes together also referred to as The Cooperative Principle.

   (a) **Maxim of Quantity:** Be as informative as is required; but not any more. The phenomenon of implicature is often an outcome of this maxim.

   (b) **Maxim of Quality:** Do not say what you don’t believe in; or what you believe in but think the evidence is insufficient.

   (c) **Maxim of Relation:** Be as relevant as possible.

   (d) **Maxim of Manner:** Be as clear, unambiguous, simple as you can while conveying the information you intend to.

### Medium Constraints

These are constraints relating to the medium of transmission between the speaker and the listener, rather than their individual states or intents, or their pairwise relationship. Nonetheless, as per the three-way SFL classification, these would constraints would be classified under interpersonal

goals, barring those which are explicitly tied to the text itself (e.g., using a maximum of 280 characters), in which case they would (also) be classified as textual goals

### Controls

A *control* is a variable or a property defined over any text, which is specified as a part of the CG to hold necessarily a particular range or subset of values for the target text. The set of all controls specified in the CG are sometimes simply referred to as *controls*. This could include for example, properties like the number of sentences, token length, or even continuous values such as entropy of the word distribution. This can also encompass properties defined by way of functions or models, such as estimators of text simplicity, or perplexity according to some pretrained language model.

### Listener

Also referred to variously as the *Addressee* or the *Target* or the *Audience*, this refers to the individual or group of people who will finally read or listen to the generated text.

### Classical NLG Pipeline (CNP) & Its Stages

Note that the CG itself can be thought of as the “zeroth stage" of this pipeline.

1. **Macroplanning:**
   This stage deals with the discourse level, i.e., the level where sentences are elements; also sometimes called the *macrostructure* or the *macro level*. This stage handles the *Content Selection* and *Content Organization* subtasks.
2. **Microplanning:**
   This stage deals with the sentence level, i.e., the level where words/phrases are elements; also sometimes called the *microstructure* or the *micro level*. This stage handles the *Sentence Aggregation*, *Lexicalization* and *Referring Expression Generation* subtasks.
3. **Surface Realization:**
   This stage handles the sole final subtask — i.e., the identically named *Surface Realization*.
4. **Need for an Overarching Rhetorical Goals Layer**
   To satisfy the rhetorical (sub) goals within the wider CG, which are by definition *not textual* in nature, the NLG model needs to potentially factor them in at each of the subtasks in the $CNP$. However, in its most basic form, the CNP only allows the CG as an input at its topmost stage, i.e., Macroplanning, from whence it can only affect the bottom stages through the content and order choices made at the topmost stage. This condition is naturally too restrictive and limiting.

It is for this reason that we introduce an overarching “Rhetorical Goals Layer" (depicted as a vertically oriented cylinder in Figure 1.5) which provides access to all rhetorical goals as well as any of their intermediate states as one traverses down the CNLP.
We borrow the idea for such a layer from [69].

### Control and Transfer tasks

Controllable generation tasks [129, 134] are generation tasks with a 2-part CG.

1. A content-based or textual goal, often simply called the *input* or *content*. For instance, in controllable infobox-to-biography generation this would simply be the Wikipedia Infobox. For each unique test-time example, the input remains fixed.
2. Ensemble of one or more content-orthogonal/non-content goals (*interpersonal* and *ideational* goals, if one follows the Systemic Functional Linguistics terminology). These goals, or the means by which these goals are achieved, are also sometimes referred to as *styles*. Example control variables for controllable infobox-to-biography generation could be audience literacy, biography length etc. The control goals can each take on two or more discrete values, and can be dynamically varied by the user at test-time.

Transfer tasks are controllable generation tasks where the *input* is a fully-realized text with an initial or default configuration of control variables. They are sometimes also referred to as *style transfer* tasks.

**Intervention Diagrams**

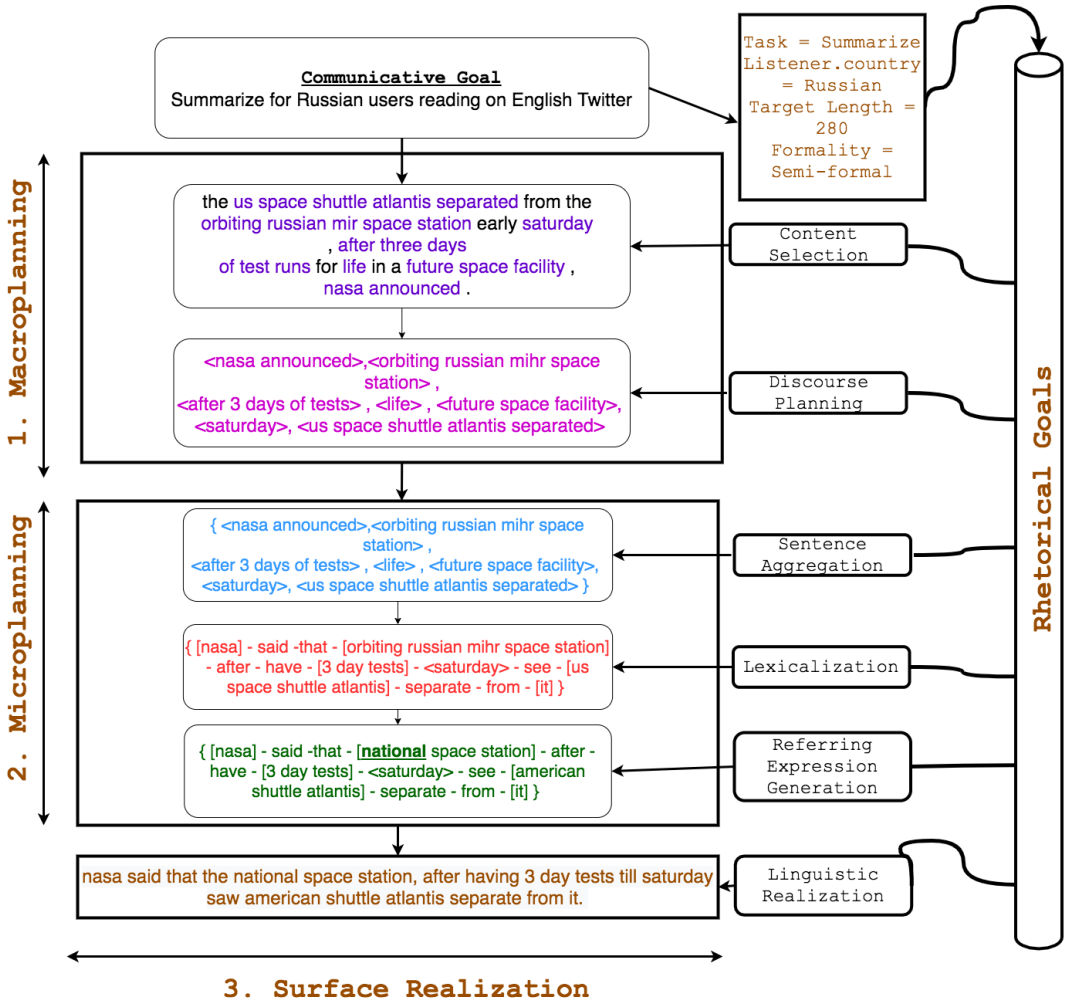



Figure 1.5: An illustration of how the *Classical NLG Pipeline* or CNP would work in action for an actual generation task and input example. Here, the task is to summarize the given input news article to within 280 characters. In addition to the classical components, we also include an overarching “Rhetorical Goals" layer, shown as a cylinder, which is seen in certain architectures such as that of [69]. The necessity of having such a layer for any reasonably realistic system is explained in §6. Having such a layer becomes a necessity for most real-world NLG tasks, since not all aspects of the communicative goal specifications deal with content (Recall the textual, ideational and interpersonal meta-function categorization from Halliday’s Systemic Functional Theory [60], which we also discuss in §5)

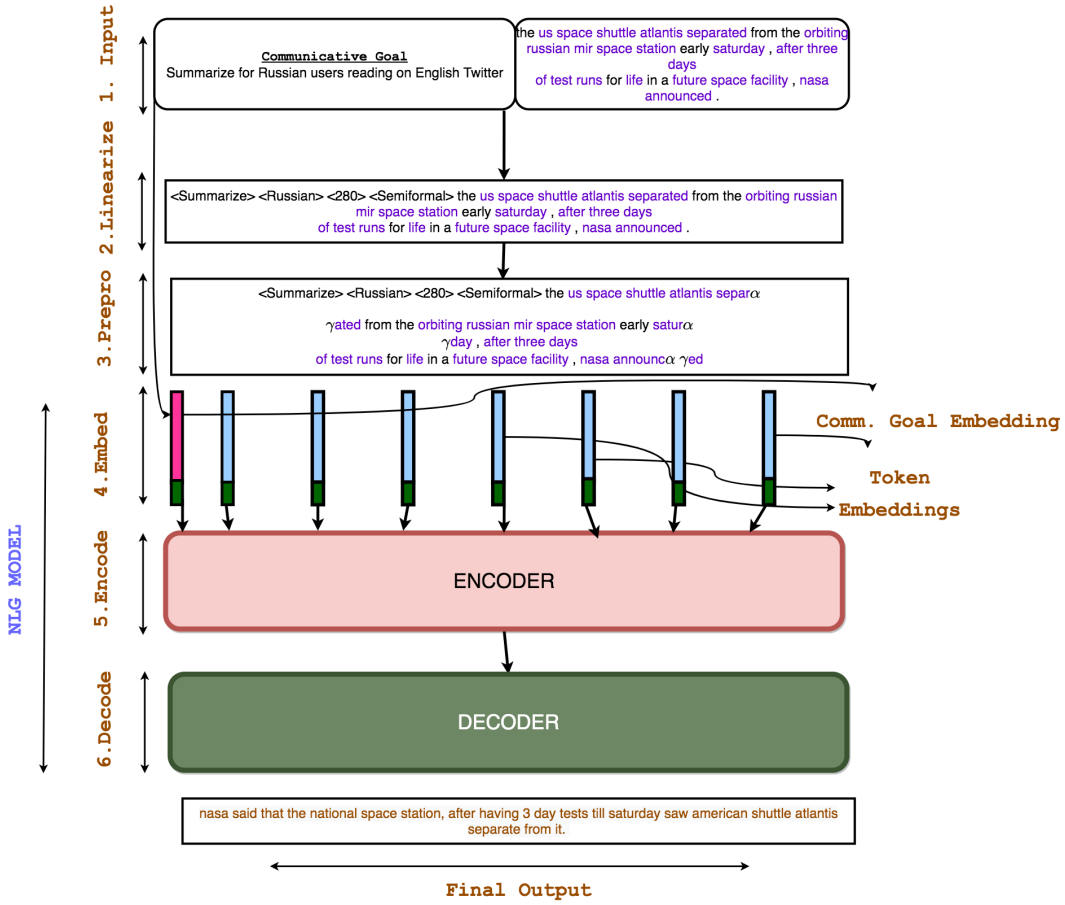



Figure 1.6: An illustration of how *End-to-End Neural NLG Pseudo-Pipeline* or E2EN2PP would work in action for an actual generation task and input example. Here, the task is to summarize the given input news article to within 280 characters. Note that this is a *Pseudo-Pipeline*, since the layers do not correspond to subtasks of NLG; moreover, they cannot be learnt or updated independently.

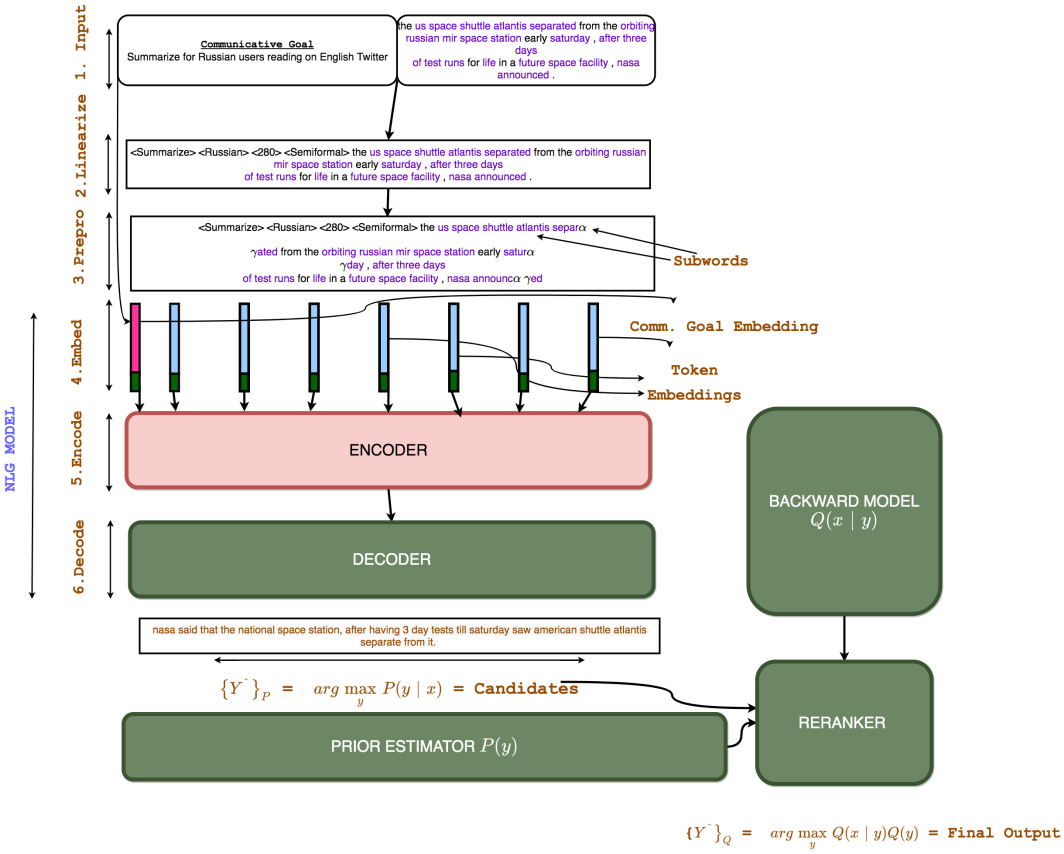



Figure 1.7: An illustration of how the *End-to-End Neural NLG Pseudo-Pipeline* or E2EN2PP fleshed out in Figure 1.6 would work in action for an actual generation task and input example, after incorporating the Intervention in Chapter 2. Here, the task is to summarize the given input news article to within 280 characters. The forward E2EN2PP here merely acts as a candidate generator, with the three new introduced components — Prior Estimator, Backward Model and Reranker producing the final output distribution used to generate the Final Output (by reranking candidates)

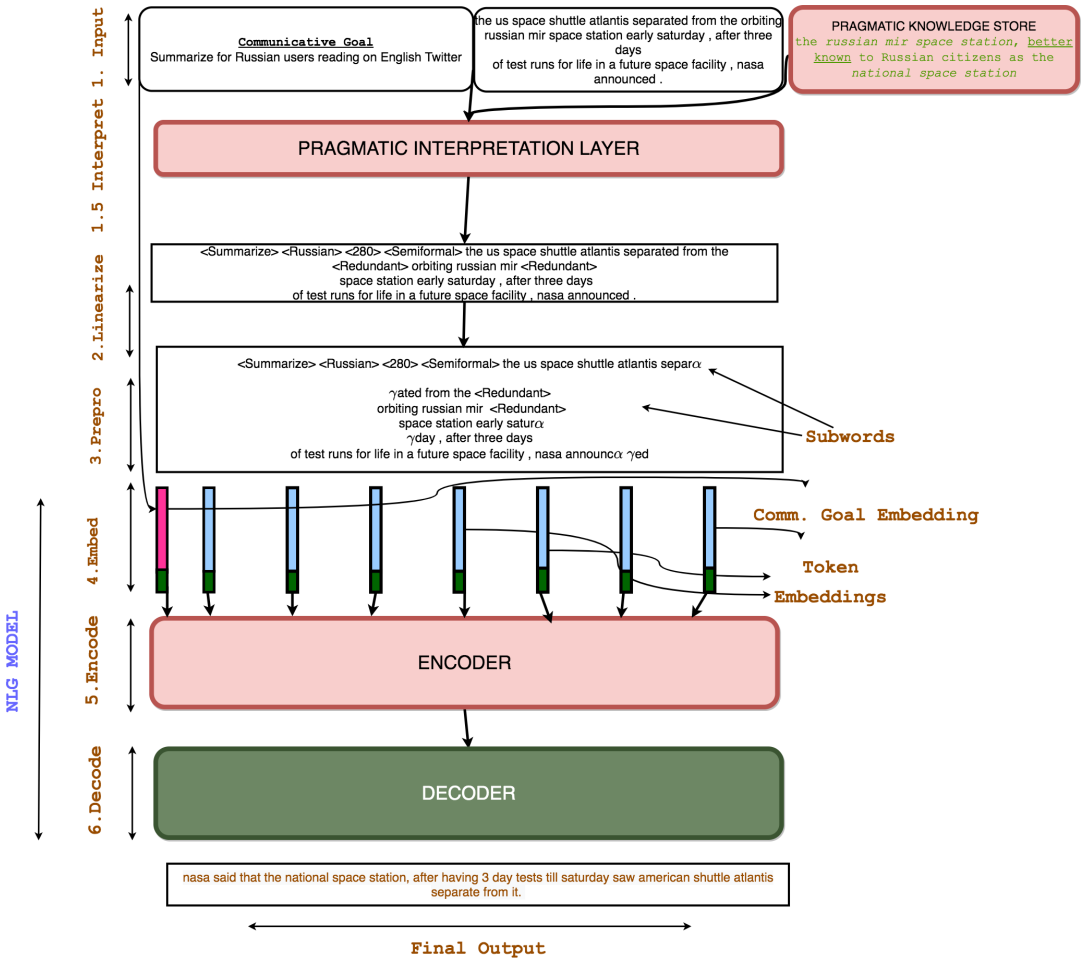



Figure 1.8: An illustration of how the E2EN2PP fleshed out in Figure 1.6 would work in action for the actual generation task and input example, after incorporating the Intervention in Chapter 7. Here, the task is to summarize the given input news article to within 280 characters. The pragmatic knowledge store here has additional knowledge about what would be apt referring expression preferences which the *Pragmatic Interpretation Layer* which it then uses to mark out redundant referring expressions which ought to be modified.

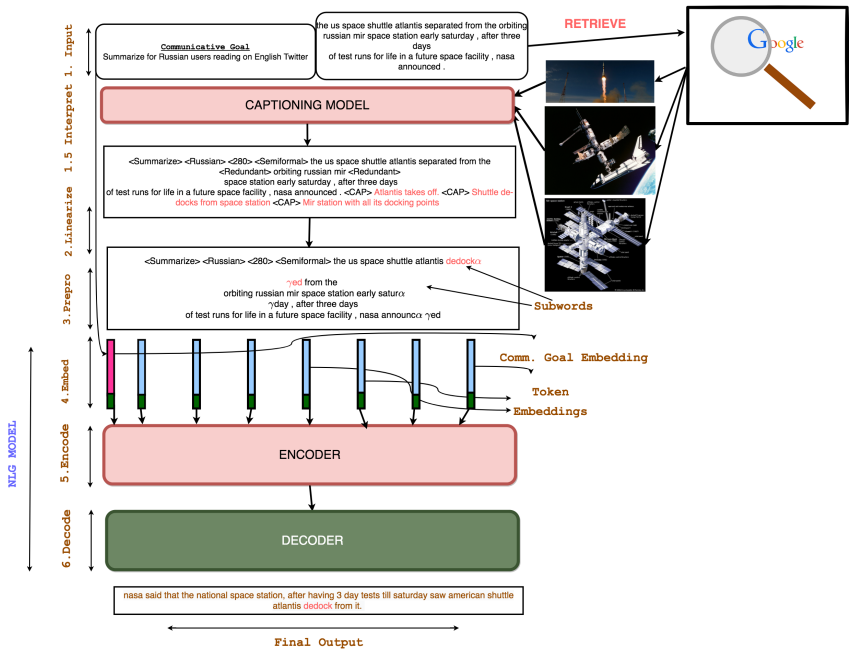



Figure 1.9: An illustration of how the E2EN2PP fleshed out in Figure 1.6 would work in action for the actual generation task and input example, after incorporating the Intervention in Chapter 6. Here, the task is to summarize the given input news article to within 280 characters. The text marked out in carrot-red in the *Final Output* , i.e., *dedocked* is clearly picked up by the model from the caption-expanded portion of the input (also marked in carrot-red)

# Part I

# Constrained Creative Settings

July 24,2022

# Chapter 2

# Portmanteau Generation (EMNLP 2017)

> For instance, take the two words "fuming" and "furious". Make up your mind that you will say both words, but leave it unsettled which you will say first . . . if you have the rarest of gifts, a perfectly balanced mind, you will say "frumious".
>
> Lewis Carroll, *Hunting Of The Snark*

Portmanteaus are a creative word formation phenomenon where two words blend to form a new word, with a meaning derived from but distinct to their original meanings e.g., *wiki* + *etiquette* → *wikiquette*, *fashion* + *fascism* → *fashism*. In this chapter, we study the **Constrained Creative** NLG setting of portmanteau generation given two root words. In terms of applications, portmanteau generation [30] can find potential use as a lower-level submodule in creative generation tasks.

The **CG Definition/Constraints** for this setting are: Given two root words, form a single word neologism i.e. a blend that is

1. Lexically and phonetically “word-like"
2. Is an effective shorthand i.e., is reminiscent of and faithful to root words

We outline and describe our **Data Availability** scenario in §2.5. We have access to only ≈ 400 examples of portmanteaus along with their root words.

We posit an underlying **Creative Story** for our setting in §2.1. Based on this creative story, we devise a two-step **Intervention** to the E2EN2PP learning process:

1. Bayesian (also known as noisy channel style) refactoring from a forward factored model P(y|x) to Noisy Channel Style model P(x|y)P(y)
2. Enforcing the “sounding word-like" bias from the CG constraint as well as the speaker’s first cognitive model from the creative story by pretraining the prior P(y) on all character sequences corresponding to English vocabulary word types.

Specifically, the intervention to the E2E NLG pipeline required here is fleshed out in Figure 2.1.

As a result of the intervention, we devise character-level, noisy channel style neural sequence-to-sequence (S2S) methods for the setting of portmanteau generation that are easily trainable, language independent, and do not explicitly use additional phonetic information. Besides the forward factored model, our experiments also find our approach to be superior to a state-of-the-art FST-based baseline with respect to ground truth accuracy and human evaluation.

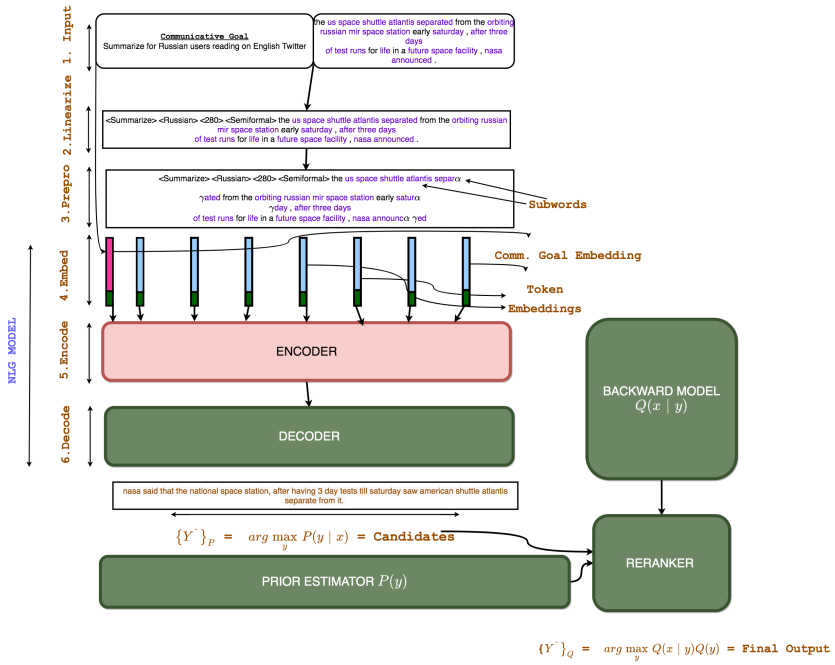



Figure 2.1: An illustration of how the *End-to-End Neural NLG Pseudo-Pipeline* or E2EN2PP fleshed out in Figure 1.6 would work in action for our actual generation task and input example, after incorporating the Intervention described in this Chapter. The forward E2EN2PP here merely acts as a candidate generator, with the three new introduced components — Prior Estimator, Backward Model and Reranker producing the final output distribution used to generate the Final Output (by reranking candidates)

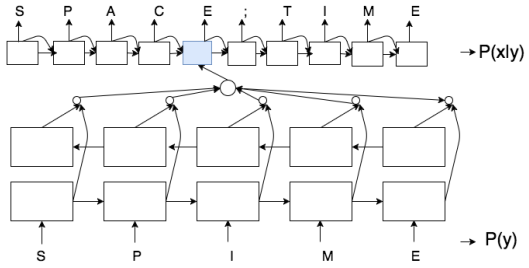



Figure 2.2: A sketch of our BACKWARD, noisy-channel model. The attentional S2S model with bidirectional encoder gives $P(x|y)$ and next-character model gives $P(y)$, where $y$ (*spime*) is the portmanteau and $\boldsymbol{x} = \text{concat}(\boldsymbol{x^{(1)}}, \text{“;”}, \boldsymbol{x^{(2)}})$ are the concatenated root words (*space* and *time*).

## 2.1 Introduction

Portmanteaus (or lexical blends [2]) are novel words formed from parts of multiple root words in order to refer to a new concept that can't otherwise be expressed concisely. Portmanteaus have become frequent in modern-day social media, news reports and advertising, one popular example being *Brexit* (Britain + Exit). [131]. These are found not only in English but many other languages such as Bahasa Indonesia [29], Modern Hebrew [7, 10] and Spanish [132]. Their short length makes them ideal for headlines and brandnames. [47].

Some languages such as Japanese also have portmanteau-like structures, albeit with fairly regular rules of formation, removing the need for a machine learning based approach. However, for English, unlike better-defined morphological phenomena such as inflection and derivation, portmanteau generation is not a typical regular phenomenon with a well-agreed upon set of rules[1]. For instance, [158] state that the composition of the portmanteau from its root words depends on several factors, two important ones being maintaining prosody and retaining character segments from the root words, especially the head.

### Creative Story

Motivated by prior work such as that of [158] described above we posit an idealized scheme for how a human speaker might construct a portmanteau given two root words. The creative story we posit involves the human speaker utilizing two internal cognitive models in unison: A first model acquired from the English vocabulary of which candidate character sequences are word-like, and

[1]This does not imply that one cannot come up with a set of rules by observing a sufficient number of portmanteau examples - but one is more likely to see a larger number of violations of these rules for newer examples than one would if portmanteaus were a regular phenomenon in the English language e.g pluralization.

a second model which checks whether the root words are “guessable" given a character sequence.

### Earlier Approaches

An existing work by [30] aims to solve the problem of predicting portmanteaus using a multi-tape FST model, which is data-driven, unlike prior approaches. Their methods rely on a grapheme to phoneme converter, which takes into account the phonetic features of the language, but may not be available or accurate for non-dictionary words, or low resource languages.

### Contributions

Prior works, such as [40], have demonstrated the efficacy of neural approaches for morphological tasks such as inflection. We hypothesize that such neural methods can (1) provide a simpler and more integrated end-to-end framework than multiple FSTs used in the previous work, and (2) automatically capture features such as phonetic similarity through the use of character embeddings, removing the need for explicit grapheme-to-phoneme prediction. To test these hypotheses, in this chapter, we devise a neural S2S model to predict portmanteaus given the two root words, specifically making 3 major contributions:

- We devise an S2S model that attends to the two input words to generate portmanteaus, and an additional improvement that leverages noisy-channel-style modelling to incorporate a language model over the vocabulary of words (§4.3.1).
- Instead of using the model to directly predict output character-by-character, we use the features of portmanteaus to exhaustively generate candidates, making scoring using the noisy channel model possible (§2.4).
- We curate and share a new and larger dataset of 1624 portmanteaus (§2.5).

In Experiments (§2.8), our model performs better than the baseline [30] on both objective and subjective measures, demonstrating that such methods can be used effectively in a morphological task.

## 2.2 Related Work

Özbal and Strapparava [123] generate new words to describe a product given its category and properties. However, their method is limited to hand-crafted rules as compared to our data driven approach. Also, their focus is on brand names. Hiranandani et al. have devised an approach to recommend brand names based on brand/product description. However, they consider only a

limited number of features like memorability and readability. Smith et al. [162] devise an approach to generate portmanteaus, which requires user-defined weights for attributes like *sounding good*. Generating a portmanteau from two root words can be viewed as a S2S problem. Recently, neural approaches have been used for S2S problems [169] such as MT. Ling et al. [103] and Chung et al. [23] have shown that character-level neural sequence models work as well as word-level ones for language modelling and MT. Zoph and Knight [193] devise S2S models for multi-source MT, which have multi-sequence inputs, similar to our case.

## 2.3 Models

This section describes our neural models.

### 2.3.1 Forward Architecture

Under our first devised architecture, the input sequence $\boldsymbol{x} = \text{concat}(\boldsymbol{x}^{(1)}, \text{“;”}, \boldsymbol{x}^{(2)})$, while the output sequence is the portmanteau $\boldsymbol{y}$. The model learns the distribution $P(\boldsymbol{y}|\boldsymbol{x})$.

The network architecture we use is an attentional S2S model [5]. We use a bidirectional encoder, which is known to work well for S2S problems with similar token order, which is true in our case. Let $\overrightarrow{LSTM}$ and $\overleftarrow{LSTM}$ represent the forward and reverse encoder; $e_{enc}()$ and $e_{dec}()$ represent the character embedding functions used by encoder and decoder. The following equations describe the model:

$$
\begin{aligned}
h_0^{\overrightarrow{enc}} &= \overrightarrow{0}, h_{|x|}^{\overleftarrow{enc}} = \overrightarrow{0} \\
h_t^{\overrightarrow{enc}} &= \overrightarrow{LSTM}(h_{t-1}^{enc}, e_{enc}(x_t)) \\
h_t^{\overleftarrow{enc}} &= \overleftarrow{LSTM}(h_{t+1}^{enc}, e_{enc}(x_t)) \\
h_t^{enc} &= h_t^{\overrightarrow{enc}} + h_t^{\overleftarrow{enc}} \\
h_0^{dec} &= h_{|x|}^{enc} \\
h_t^{dec} &= LSTM(h_{t-1}^{dec}, [\text{concat}(e_{dec}(y_{t-1}), c_{t-1})]) \\
p_t &= \text{softmax}(W_{hs}[\text{concat}(h_t^{dec}, c_t)] + b_s)
\end{aligned}
$$

The context vector $c_t$ is computed using dot-product attention over encoder states. We choose dot-product attention because it doesn't add extra parameters, which is important in a low-data scenario such as portmanteau generation.

$$a_i^t = dot(h_t^{dec}, h_i^{enc}), \alpha^t = \text{softmax}(a^t)$$
$$c_t = \sum_{i=1}^{i=|x|} \alpha_i^t h_i^{enc}$$

In addition to capturing the fact that portmanteaus of two English words typically sound *English-like*, and to compensate for the fact that available portmanteau data will be small, we pretrain the character embeddings on English language words. We use character embeddings learnt using an LSTM language model over words in an English dictionary,[2] where each word is a sequence of characters, and the model will predict next character in sequence conditioned on previous characters in the sequence.

### 2.3.2 Backward Architecture

The second devised model uses Bayes's rule to reverse the probabilities $P(\boldsymbol{y}|\boldsymbol{x}) = \frac{P(\boldsymbol{x}|\boldsymbol{y})P(\boldsymbol{y})}{P(\boldsymbol{x})}$ to get $\text{argmax}_{\boldsymbol{y}}\, P(\boldsymbol{y}|\boldsymbol{x}) = \text{argmax}_{\boldsymbol{y}}\, P(\boldsymbol{x}|\boldsymbol{y})P(\boldsymbol{y})$. Thus, we have a reverse model of the probability $P(\boldsymbol{x}|\boldsymbol{y})$ that the given root words were generated from the portmanteau and a character language model $P(\boldsymbol{y})$. This is a probability distribution over all character sequences $y \in A^*$, where $A$ is the alphabet of the language. This way of factorizing the probability is also known as a *noisy channel model*, which has recently also been shown to be effective for neural MT ([65], [187]). Such a model offers two advantages

1. The reverse direction model (or alignment model) gives higher probability to those portmanteaus from which one can discern the root words easily, which is one feature of good portmanteaus.
2. The character language model $P(\boldsymbol{y})$ can be trained on a large vocabulary of words in the language. The likelihood of a word $y$ is factorized as $P(y) = \Pi_{i=1}^{i=|y|} P(y_i|y_1^{i-1})$, where $y_j^i = y_i, y_{i+1} \ldots y_j$, and we train a LSTM to maximize this likelihood.

## 2.4 Making Predictions

Given these models, we must make predictions, which we do by two methods

[2]Specifically in our experiments, 134K words from the CMU Pronouncing dictionary [181].

**Greedy Decoding:** In most neural sequence-to-sequence models, we perform auto-regressive greedy decoding, selecting the next character greedily based on the probability distribution for the next character at current time step. We refer to this decoding strategy as GREEDY.

**Exhaustive Generation:** Many portmanteaus were observed to be concatenation of a prefix of the first word and a suffix of the second. We therefore generate all candidate outputs which follow this rule. Thereafter we score these candidates with the decoder and output the one with the maximum score. We refer to this decoding strategy as SCORE.

Given that our training data is small in size, we expect ensembling [15] to help reduce model variance and improve performance. In this chapter, we ensemble our models wherever mentioned by training multiple models on 80% subsamples of the training data, and averaging log probability scores across the ensemble at test-time.

## 2.5 Dataset

The existing dataset by [30] contains 401 portmanteau examples from Wikipedia. We refer to this dataset as $D_{Wiki}$. Besides being small for detailed evaluation, $D_{Wiki}$ is biased by being from just one source. We manually collect $D_{Large}$, a dataset of 1624 distinct English portmanteaus from following sources:

- Urban Dictionary[3]
- Wikipedia
- Wiktionary
- BCU's Neologism Lists from '94 to '12.

Naturally, $D_{Wiki} \subset D_{Large}$. We define $D_{Blind} = D_{Large} - D_{Wiki}$ as the dataset of 1223 examples not from Wikipedia. We observed that 84.7% of the words in $D_{Large}$ can be generated by concatenating prefix of first word with a suffix of the second.

## 2.6 Baseline

In this section, we concisely discuss the FST-based approach for our task, devised by [30]. We refer the reader to the original paper for a more detailed exposition.

The baseline approach from [30], illustrated in Figure 2.3, defines a pipeline of FSTs to progressively transform the root words $\boldsymbol{x^{(1)}}$ and $\boldsymbol{x^{(2)}}$ to the output $\boldsymbol{y}$. It also requires the root

[3]Not all neologisms are portmanteaus, so we manually choose those which are for our dataset.

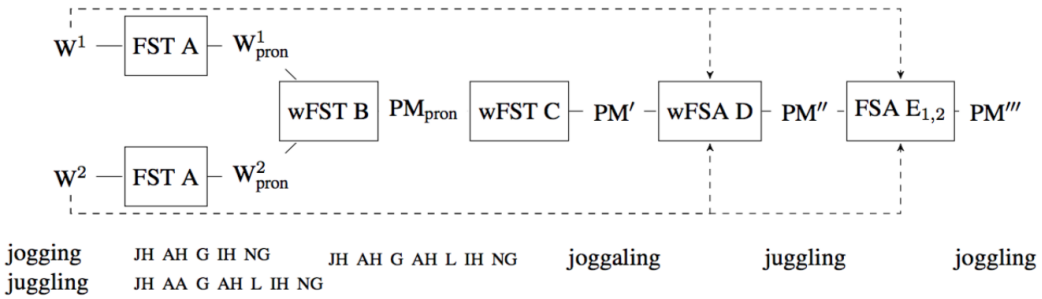



Figure 2.3: A sketch of the BASELINE FST-based pipeline approach from [30], starting with the input rood words *jogging* and *juggling* to the left, leading to the final output, *joggling* at the rightmost end. This approach requires both root words $\boldsymbol{x}^{(1)}$ and $\boldsymbol{x}^{(2)}$ to be present in the CMU Phonetic Dictionary to get the phonetic sequences for the first step, as shown.

words to have corresponding phoneme sequences based on the CMU Pronouncing Dictionary. The approach proceeds through the following steps.

1. Get the phoneme sequences of root words $\boldsymbol{x}^{(1)}$ and $\boldsymbol{x}^{(2)}$ from CMU Pronouncing Dictionary
2. FST $A$ pretransforms the individual phoneme sequences before they are merged.
3. FST $B$, which has two input tapes (one for each root word) generates a merged phoneme sequence on its "output tape", denoted by $PM_{pron}$.
4. FST $C$ reads off this phoneme sequence and reconverts it to the space of graphemes/letters, with the output grapheme sequence being denoted $PM'$.
5. Finally, FST $D$ and FST $E_{1,2}$ each do a round of successive "post-processing" of sorts, converting $PM' \rightarrow PM''$ and $PM' \rightarrow PM''$ respectively. FST $E_{1,2}$ has three input tapes, since it also reads in the two root words as inputs in addition to the current portmanteau sequence $PM''$.
6. The final, output portmanteau produced by FST is denoted $PM'''$.

The first disadvantage of this approach is its dependence on converting root words to their phoneme sequences explicitly, which requires their presence in the CMU Pronouncing Dictionary. Though this is not an issue for root words in the $D_{Wiki}$, which are all covered by the dictionary, we find 6.36% of the root words in $D_{Blind}$ missing from CMU Phonetic Dictionary.

## 2.7 Evaluation Measures

We assess models automatically using two distinct evaluation measures.

1. *Matches*: This is a 0-1 (i.e., binary/boolean) measure that checks whether, for a given example,

| Model | Attn | Ens | Init | Search | Matches | Distance |
|---|---|---|---|---|---|---|
| BASELINE | - | - | - | - | **45.39%** | 1.59 |
| FORWARD | ✓ | × | × | GREEDY | 22.00% | 1.98 |
| | ✓ | × | ✓ | GREEDY | 28.00% | 1.90 |
| | ✓ | × | × | BEAM | 13.25% | 2.47 |
| | ✓ | × | ✓ | BEAM | 15.25% | 2.37 |
| | ✓ | × | × | SCORE | 30.25% | 1.64 |
| | ✓ | × | ✓ | SCORE | 32.88% | 1.53 |
| | ✓ | ✓ | ✓ | SCORE | 42.25% | 1.33 |
| | ✓ | ✓ | × | SCORE | 41.25% | 1.34 |
| | × | × | ✓ | SCORE | 6.75% | 3.78 |
| | × | × | × | SCORE | 6.50% | 3.76 |
| BACKWARD | ✓ | × | × | SCORE | 37.00% | 1.53 |
| | ✓ | × | ✓ | SCORE | 42.25% | 1.35 |
| | ✓ | ✓ | ✓ | SCORE | **48.75%** | **1.12** |
| | ✓ | ✓ | × | SCORE | **46.50%** | **1.24** |
| | × | × | ✓ | SCORE | 5.00% | 3.95 |
| | × | × | × | SCORE | 4.75% | 3.98 |

Table 2.1: 10-Fold Cross-Validation results, $D_{Wiki}$. *Attn, Ens, Init* denote attention, ensembling, and initializing character embeddings respectively.

the string form of the predicted output $y$ exactly matches that of the gold output $y_{gold}$.

2. *Distance*: This measure corresponds to the Levenshtein edit distance [94] between the string form of the predicted output $y$ and that of the gold output $y_{gold}$. Levenshtein edit distance between two strings $a$ and $b$ is defined as the shortest sequence of character edits (insertions+deletions+replacements) needed to transform $a$ into $b$. Intuitively, this allows a relaxed, smoother comparison between $y$ and $y_{gold}$ compared to *Matches*. Furthermore, this allows more sensitivity to some divergence from $y_{gold}$, allowing a candidate output to receive partial credit for resembling but not exactly matching $y_{gold}$.

## 2.8 Experiments

In this section, we show results comparing various configurations of our model to the baseline FST model of [30] (BASELINE). Models are evaluated using exact-matches (*Matches*) and average Levenshtein edit-distance (*Distance*) w.r.t. ground truth.

### 2.8.1 Objective Evaluation Results

In *Experiment 1*, we follow the same setup as [30]. $D_{Wiki}$ is split into 10 folds. Each fold model uses 8 folds for training, 1 for validation, and 1 for test. The average (10 fold cross-validation style approach) performance metrics on the test fold are then evaluated. *Table 5.3* shows the results of *Experiment 1* for various model configurations. We get the BASELINE numbers from [30]. Our best model obtains $48.75\%$ *Matches* and $1.12$ *Distance*, compared to $45.39\%$ *Matches* and $1.59$ *Distance* using BASELINE.

| Model | Attn | Ens | Init | Search | Matches | Distance |
|---|---|---|---|---|---|---|
| BASELINE | - | - | - | - | **31.56%** | 2.32 |
| FORWARD | ✓ | × | ✓ | SCORE | 25.26% | 2.13 |
| | ✓ | × | × | SCORE | 24.93% | 2.32 |
| | ✓ | ✓ | × | SCORE | 31.23% | 1.98 |
| | ✓ | ✓ | ✓ | SCORE | 28.94% | 2.04 |
| BACKWARD | ✓ | × | ✓ | SCORE | 25.75% | 2.14 |
| | ✓ | × | × | SCORE | 25.26% | 2.17 |
| | ✓ | ✓ | × | SCORE | **31.72%** | **1.96** |
| | ✓ | ✓ | ✓ | SCORE | **32.78%** | **1.96** |

Table 2.2: Results on $D_{Blind}$ (1223 Examples). In general, BACKWARD architecture performs better than FORWARD architecture.


Figure 2.4: Attention matrices while generating *slurve* from *slider;curve*, and *bennifer* from *ben;jennifer* respectively, using *Forward* model. **;** and **.** are separator and stop characters. Darker cells are higher-valued

For *Experiment 2*, we seek to compare our best approaches from *Experiment 1* to the BASELINE on a large, held-out dataset. Each model is trained on $D_{Wiki}$ and tested on $D_{Blind}$. BASELINE was similarly trained only on $D_{Wiki}$ , making it a fair comparison. Table 2.2 shows the results[4]. Our best model gets *Distance* of 1.96 as compared to 2.32 from BASELINE.

We observe that the *Backward* architecture performs better than *Forward* architecture, confirming our hypothesis in §2.3.2. In addition, ablation results confirm the importance of attention, and initializing the word embeddings. We believe this is because portmanteaus have high fidelity towards their root word characters and its critical that the model can observe all root sequence characters, which attention manages to do as shown in Fig. 5.2.

[4]For BASELINE [30], we use the model from http://leps.isi.edu/fst/step-all.php

**Performance on Uncovered Examples**

The set of candidates generated before scoring in the approximate SCORE decoding approach sometimes do not include the ground truth. Some such uncovered examples are *precise+exactly* $\rightarrow$ *prezactly* and *puke+extravaganza* $\rightarrow$ *pukestravaganza*. This holds true for 229 out of 1223 examples in $D_{Blind}$. We compare the FORWARD approach along with a GREEDY decoding strategy to the BASELINE approach for these examples.

Both FORWARD+GREEDY and the BASELINE get 0 *Matches* on these examples. The *Distance* for these examples is 4.52 for BASELINE and 4.09 for FORWARD+GREEDY. Further, a spot checked inspection for a randomly chosen subsample also confirms example outputs from both the approaches to be of comparable quality. Hence, we see that one of our approaches (FORWARD+GREEDY) stands at par with the BASELINE even for these examples.

### 2.8.2 Significance Tests

Since our dataset is still small relatively small (1223 examples), it is essential to verify whether BACKWARD is indeed statistically significantly better than BASELINE in terms of *Matches*.

In order to do this, we use a paired bootstrap[5] comparison [89] between BACKWARD and BASELINE in terms of *Matches*. BACKWARD is found to be better (gets more *Matches*) than BASELINE in 99.9% ($p = 0.999$) of the subsets.

Similarly, BACKWARD has a lower *Distance* than BASELINE by a margin of $0.2$ in 99.5% ($p = 0.995$) of the subsets.

### 2.8.3 Subjective Evaluation and Analysis

On inspecting outputs, we observed that often output from our system seemed good in spite of high edit distance from ground truth. Such aspect of an output *seeming good* is not captured satisfactorily by measures like edit distance. To compare the errors made by our model to the baseline, we designed and conducted a human evaluation task on AMT.[6] In the survey, we show human annotators outputs from our system and that of the baseline. We ask them to judge which alternative is *better* overall based on following criteria: 1. It is a good shorthand for two original words 2. It sounds better. We requested annotation on a scale of 1-4. To avoid ordering bias, we shuffled the order of two portmanteau between our system and that of baseline. We restrict

[5]We average across $M = 1000$ randomly chosen subsets of $D_{Blind}$, each of size $N = 611$ ($\approx 1223/2$)

[6]We avoid ground truth comparison because annotators can be biased to ground truth due to its existing popularity.

| Input | FORWARD | BACKWARD | BASELINE | G.TRUTH |
|---|---|---|---|---|
| shopping;marathon | shopparathon | shoathon | shon | shopathon |
| fashion;fascism | fashism | fashism | fashism | fashism |
| wiki;etiquette | wikiquette | wiquette | wiquette | wikiquette |
| clown;president | clowident | clownsident | clownt | clownsident |
| car;hijack | carjack | carjack | cack | carjack |
| dialectical;materialism | dialerialism | dialerialism | dialism | dialerialism |
| tinder;disaster | tinter | tindersaster | tindisaster | tindisaster |
| data;broadcasting | datasting | doadcasting | dating | datacasting |
| back;acronym | backronym | bacronym | bacronym | backronym |
| britain;regret | bregret | brigret | bregret | bregret |
| social;entertainer | socialtainer | sociartainer | sentertainer | socialtainer |
| chopstick;fork | chopstork | chopfork | chork | chork |
| happy;harmonius | happonius | happonius | hharmonius | happymonius |
| flexible;vegetarian | flexarian | flexetarian | flegetarian | flexitarian |
| laughter;orgasm | lauggasm | laughtergasm | lasm | laughgasm |
| frequency;recency | frecency | frecency | frecency | frecency |
| tender;enterpreneur | tenpreneur | tendereneur | tenterpreneur | tenderpreneur |
| fall;halloween | falloween | falloween | falloween | falloween |
| frisbee;golf | frolf | frisbolf | frolf | frolf |
| hitler;hillary | hitlary | hitlery | hitlery | hitlery |
| trump;economics | trumpics | trumponomics | trumics | trumponomics |
| flirtation;relationship | flirtionship | flirtationship | flirtationship | flirtationship |
| next;yesterday | nexterday | nesterday | nexterday | nexterday |
| lobster;monstrosity | lobstrosity | lonstrosity | lobstrosity | lobstrosity |
| global;english | glonglish | globlish | glish | globlish |
| puke;extravaganza | pukaganza | pukaganza | puextravaganza | pukestravaganza |
| beverage;avalanche | bevalanche | beveranche | bavalanche | bevelanche |
| excited;dimmer | excimmer | excimmer | excitedimmer | excimmer |
| phone;amnesia | phonesia | phonesia | phonesia | phonesia |
| camera;phone | came | camphone | camphone | camphone |
| bored;ordinary | bordinary | bordinary | bordinary | bordinary |
| precise;exactly | prexactly | prexactly | prexactly | prezactly |

Table 2.3: Example outputs from different models. Outputs are from best peforming configurations of the models. G.TRUTH denotes the ground truth portmanteau.

annotators to be from Anglophone countries, have HIT Approval Rate $> 80\%$ and pay $0.40$\$ per HIT (5 Questions per HIT). We had 2 annotations per question and considered each annotation as a separate judgement, without doing per-example aggregation. Nevertheless, the two annotations showed a reasonably high Pearson correlation of $0.6191$.

As seen in Table 2.4, output from our system was labelled better by humans as compared to the baseline 58.12% of the time. Table 2.3 shows outputs from different models for a few examples.

| Judgement | Percentage of total |
|---|---|
| Much Better (1) | 29.06 |
| Better (2) | 29.06 |
| Worse (3) | 25.11 |
| Much Worse (4) | 16.74 |

Table 2.4: AMT annotator judgements on whether our system's proposed portmanteau is better or worse compared to the baseline

## 2.9 Conclusion

In this chapter, we explored a data-deficient setting involving NLG as a constrained creative process at the lexical and phonetic level.

We had access to only $\approx 400$ examples of portmanteaus along with their root words. Thus, we had the poor Data Availability characteristic of this class of settings being present.

We first hypothesized an underlying Creative Story (see §2.1 for more). Based on this creative story, we devised an Intervention to the typical E2EN2PP learning process to change the learning process in two step. The first step involved a Bayesian refactoring to make the model distribution structure closer to the Creative Story. The second involved an unsupervised pretraining step to better learn the prior component of this distribution.

Specifically, we devised a noisy channel style, neural NLG system to model portmanteau generation. Our experiments show the efficacy of devised system in predicting portmanteaus given the root words. Our methods can be learnt without using external phonetic resources, and learn from existing list of portmanteaus and word types.

We conclude that pretraining character embeddings on the English vocabulary helps the model. When we additionally incorporate a character-level next-character prediction module pretrained on word types through a prior, we are able to outperform both:
a) earlier state-of-the-art models based on explicit phonetic knowledge from [30].
b) Non-Bayesian, forward factored and non-pretrained "direct" neural approaches.

This shows that the Intervention performed to the E2EN2PP as shown in Figure 2.1, leading to a departure from its end-to-end nature, was indeed justified and benefited end-task performance.

Through human evaluation we show that our model's predictions compare favourably to the baseline. We have also released our dataset and code[7] to encourage further research on the phenomenon of portmanteaus. An obvious extension to our work is to try similar models on multiple languages.

[7] https://github.com/vgtomahawk/Charmanteau-CamReady

# Chapter 3

# Personification Generation

## (Under Review @ COLING 2022)

> As for me —there is another partner waiting for me, a teacher whom I knew long ago — his name is solitude.
>
> Iris Murdoch — *The Green Knight*

> Know the grave doth gape for thee thrice wider than for other men.
>
> William Shakespeare — *Henry IV*

A personification is a figure of speech that endows inanimate entities with properties and actions typically seen as requiring animacy. In this chapter, we study the **Constrained Creative** NLG setting of personification generation. To this end, we formulate **PINEAPPLE**: **P**ersonifying **IN**animate **E**ntities by **A**cquiring **P**arallel **P**ersonification Data for **L**earning **E**nhanced Generation.

The **CG Definition/Constraints** for this setting are: Given a literal source sentence lacking prior personification, introduce personification by assigning an inanimate entity animacy-requiring attributes/roles e.g., *My alarm clock yells at me every morning*, where the inanimate *alarm clock* is assigned as the agent of the animate agent-requiring verb *yells*. This CG constraint points to the importance of the deeper layer of underlying dependency structure in the target-side, personified sentences.

We outline and describe in detail our **Data Availability** scenario in §3.2. As training data, we only have access to $\approx$ 350 examples of target-side, personified sentences. Note that this data,

besides being low count, is also incomplete in terms of the typical source-target parallel data used to train end-to-end models This is since we lack access to source-side, depersonified input sentences.

We posit an underlying **Creative Story** for our setting in §3.2.1, inspired by similar theories underlying the creation of similes. Based on this creative story, we devise a two-step intervention to the E2EN2PP corpus curation and consequently, the overall training process.

1. Devising a “de-personification" pipeline in §3.2.2 to construct noisy source-side inputs $x_{noisy}$ which replace the animacy-requiring portions of the sentence with equivalent animacy-agnostic ones. using off-the-shelf dependency parsing, commonsense knowledge bases and pre-trained BART infilling followed by a 3-component candidate heuristic to enforce loss of ATTRIBUTE animacy, whilst preserving fluency and meaning
2. Using the constructed, pseudo-parallel pairs to finetune a well-pretrained transduction model checkpoint, e.g., BART or T5.

Specifically, the intervention to the E2EN2PP corpus curation and overall training process required here is fleshed out through Figures 3.3 and 1.2 respectively. The de-personification pipeline shown in Figure 3.3 is in turn based on the underlying TOPIC-ATTRIBUTE schematic model/“creative story".

We demonstrate the usefulness of our pseudo-parallel corpus by training a seq2seq model to personify a given literal input. Both automatic and human evaluations show that fine-tuning with *PersonifCorp* leads to significant gains in personification-related qualities such as animacy and interestingness. A detailed qualitative analysis also highlights key strengths and imperfections of **PINEAPPLE** over baselines, demonstrating a strong ability to generate diverse and creative personifications that enhance the overall appeal of a sentence.

## 3.1 Introduction

Personification is the attribution of animate actions or characteristics to an entity that is inherently inanimate. Consider, for example, the sentence *“The **stars danced playfully** in the moonlit sky.”* Here, the vibrance of the stars (something inanimate) is being likened to dancing playfully, which is a distinctly animate action. By allowing readers to construct clearer mental images, personifications enhance the creativity of a piece of text [12, 33, 45].

Being able to automatically identify and generate personifications is important for multiple reasons. First, humans naturally use personifications when communicating. When we say

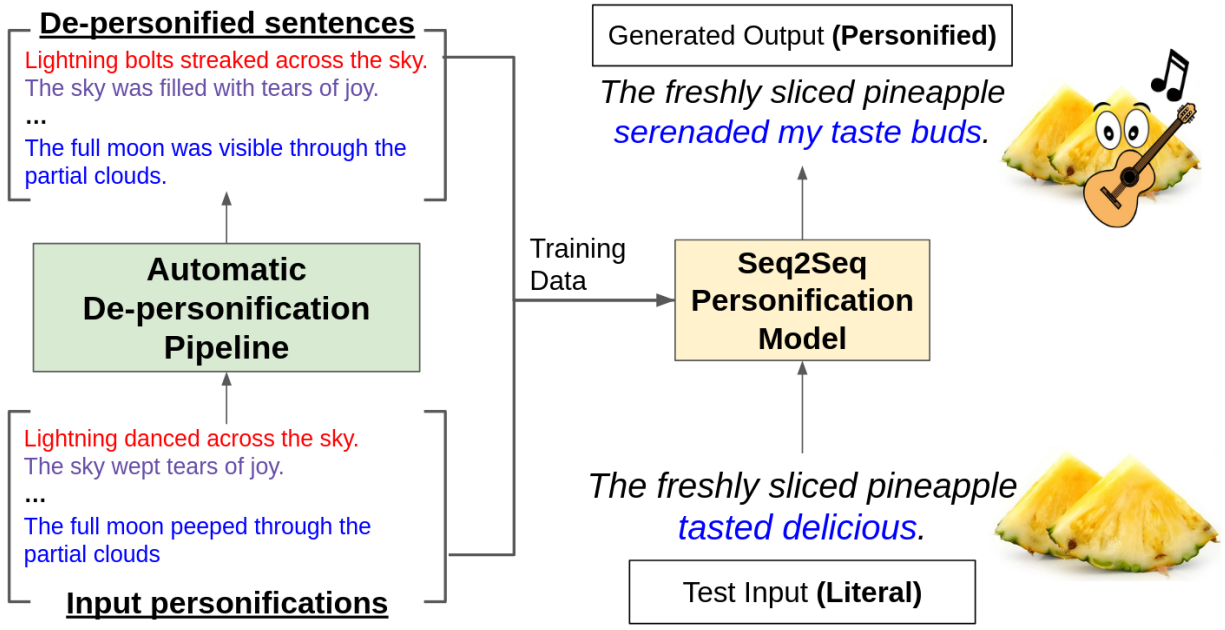



Figure 3.1: Overall **PINEAPPLE** model pipeline. The left part of the diagram shows the corpus creation process, while the right part of the diagram shows the training and generation process.

something like *"My phone has died,"* or *"My car is not cooperating,"* to a dialogue system, it is important that the dialogue system understands the intended meaning behind these personifications. If these systems interpret personifications literally, they may fail in several downstream tasks (e.g., classification) since their understanding is incorrect. Being able to generate personifications also allows dialogue agents and language models to be more creative and generate more figurative sentences. Personification generation has additional applications such as AI-assisted creative writing, since machine-generated figures of speech have been shown to enhance the interestingness of written text [19].

Despite previous success in generating other figures of speech such as similes [18], metaphors [166], hyperboles [173], irony [174], and sarcasm [61, 71], personification generation is relatively underexplored. One key challenge is that personifications do not have an explicit syntactic structure unlike similes which use *'like'* or *'as'*. They are also not as loosely-defined as metaphors. Rather, a personification requires identifying an inanimate subject together with actions or descriptions which are commonly used on animate subjects. These steps are challenging and require our models to understand commonsense concepts including animacy.

In line with exploring the task of personification generation, we present three main contributions: (1) We curate a dataset, *PersonifCorp*, of diverse personification examples from various

sources. (2) We formulate a method called **PINEAPPLE** to automatically de-personify personifications and create a parallel corpus of personification data along with their literalizations. (3) Given our parallel corpus, we train a seq2seq model to personify given text. We conduct automatic and human evaluation and qualitative analysis of the generated outputs.

## 3.2 Datasets

We curate a dataset called *PersonifCorp* of 511 personifications, with 236 coming from a publicly available open-sourced list[1] and 275 manually-filtered personifications extracted from the Deja dataset [22]. The Deja dataset is an image-captioning dataset containing a "figurative" subset of size 6000, of which $4.1\%$ of the captions are labelled as personifications. We extract these personifications and combine them with our existing list to form the final *PersonifCorp* dataset.

### 3.2.1 Characterizing Personifications

We define the *elements of personification*, an analogue to what was previously done for similes [18, 119]. While similes could be decomposed into very granular structures and well-defined elements, the unstructured nature of personifications prevents us from directly defining such fine-grained elements for personifications. Rather, we define two main high-level elements, the TOPIC (a noun phrase that acts as logical subject) and the ATTRIBUTE (the distinctly animate action or characteristic that is being ascribed to the TOPIC). Figure 3.2 shows examples of how these TOPICS and ATTRIBUTES can relate to each other.

### 3.2.2 Automatic Parallel Corpus Construction

In order to train a seq2seq model to generate high-quality personifications, we need pairs of personifications along with their corresponding literalizations. However, the literalization process may take several human-hours, which is impractical for large datasets. We therefore formulate **PINEAPPLE**, a three-stage automatic de-personification process, where we first identify all valid TOPIC-ATTRIBUTE pairs, then generate multiple candidates to replace the ATTRIBUTE of each TOPIC. Lastly, we select the most appropriate candidate in terms of animacy, fluency, and meaning preservation. These steps are further detailed individually below:

[1] https://www.kaggle.com/datasets/varchitalalwani/figure-of-speech

| ATTRIBUTE Type | Example |
|---|---|
| Noun | The planet earth is our mother. |
| Verb | My alarm clock yells at me to get out of bed every morning. |
| Adjective | Justice is blind and, at times, deaf. |

Figure 3.2: Examples of different types of personification ATTRIBUTES (TOPICS in red and ATTRIBUTES in blue).

**TOPIC-ATTRIBUTE Extraction.** To identify the TOPICS and ATTRIBUTES, we consider the dependency parse tree of a sentence and the part-of-speech (POS) tags of each of its words. Given the tree, we extract all the nouns/pronouns which have edges pointing into it with the *nominal subject* label, together with the corresponding parent nodes. For instance, in the sentence *"The stars danced in the night sky"*, the word *'danced'* is a parent of the word *'stars'*, with the *nominal subject* edge relationship. We can thus identify *'stars'* as the TOPIC and *'danced'* as the ATTRIBUTE. In more complex scenarios, we may need to perform some additional merging to deal with compound multi-word TOPICS and ATTRIBUTES, as well as any additional modifiers. More specifically, using the POS tags, we identify all words tagged as *negation modifiers*, *possession modifiers*, *nominal modifiers*, *adjectival complements*, and *objects of prepositions*, and words tagged as determiners and parts of compound phrases.[2] After extracting these nodes, they are iteratively merged with their parents in the dependency parse tree, and the merging process is performed repeatedly until no more merges are possible. The final TOPIC-ATTRIBUTE pairs are then identified using the *nominal subject* edge relationship as previously described. Examples of the merging process can be found in Appendix 3.7.1.

**Candidate Generation.** Once the TOPIC-ATTRIBUTE pairs have been identified, we then determine which TOPICS are inanimate. To achieve this, we need some type of commonsense notion of what constitues animacy. We use COMET [14] to tap into the commonsense knowledge present in large-scale knowledge graphs such as ConceptNet [163]. Although ConceptNet has no explicit notion of animacy, it has certain edge relations that we can leverage to design a proxy metric. More specifically, we use the *IsA* relation to design a custom *IsAPerson* animacy metric. If the TOPIC of our sentence refers to an animate entity, then we expect its *IsA* relation score with

[2]The spaCy library was used to extract the dependency tree and POS tags.

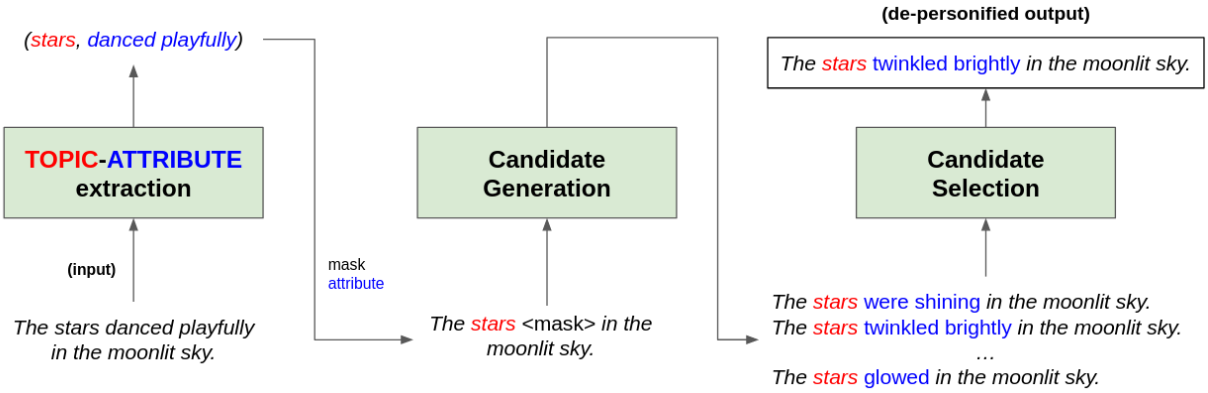



Figure 3.3: Overview of the **PINEAPPLE** de-personification pipeline.

| **Original Personification** | **Result After De-Personifying** |
|---|---|
| How far that little candle throws its beams! | How far that little candle can spread its beams! |
| A book is a fragile creature, it suffers the wear of time, it fears rodents, the elements and clumsy hands. | A book is fragile, it can break from the wear of time, it can be eaten by rodents, the elements and clumsy hands. |
| The camera loves her since she is so pretty. | The camera is always on her since she is so pretty. |
| Any trust I had for him walked right out the door. | Any trust I had for him had gone right out the door. |
| The full moon peeped through partial clouds. | The full moon was visible through partial clouds. |
| Opportunity was knocking at her door. | Opportunity was knocking at her door. |
| The killing moon will come too soon. | The killing moon will be here too soon. |

Table 3.1: Example outputs of the **PINEAPPLE** de-personification pipeline. The ATTRIBUTES are highlighted in blue for both the original personifications, as well as the de-personified output sentences. The last two rows contain negative examples where the process does not successfully de-personify the input.

the word *'human'* to be relatively low.[3] The *IsAPerson* metric is hence defined as follows: given a TOPIC, we compute and average its *IsA* scores to various words that are synonymous or very closely related to *'human'*, such as *'person'*, *'man'*, and *'woman'*. We call this set of *'human'*-related words the HUMANSET. The construction and full list of words in the HUMANSET can be found in Appendix 3.7.2. The average of these ConceptNet scores is then our final *IsAPerson* animacy score.

Phrases whose *IsAPerson* animacy score exceeds a certain threshold [4] are considered animate;

[3]For the COMET ConceptNet graph, lower scores correspond to better matches.

[4]We use a threshold of 7.0 for the *IsAPerson* animacy metric. *IsAPerson* scores $< 7.0$ are considered animate, while scores $\geq 7.0$ are considered inanimate. More details regarding the selection of this threshold can be found in Appendix 3.7.3.

otherwise, they are considered inanimate. Since our goal is to de-personify a sentence, we can safely discard all the animate TOPICS, as these need no further de-personification. Rather, we focus on the inanimate TOPICS because the segment we want to de-personify most likely occurs in the TOPIC-ATTRIBUTE pairs whose TOPIC is inanimate. Once we identify all such inanimate TOPIC-ATTRIBUTE pairs, we mask out the ATTRIBUTE of each of them with `<mask>`, then use a pre-trained BART model [96] to generate the top $k = 10$ candidates for each mask using beam search with a beam size of 10. The goal of this process is to replace a possibly animate action/characteristic with candidates that are inanimate.

**Candidate Selection.** Given $k = 10$ candidate replacement ATTRIBUTES, we now select the most ideal replacement based on three metrics: animacy, fluency, and meaning preservation.

1. Animacy – We want the replacement ATTRIBUTE to be inanimate; otherwise we would just be replacing an animate ATTRIBUTE with another animate ATTRIBUTE. We define the animacy of a TOPIC-ATTRIBUTE pair as difference between the affinity for a human ($\mathcal{A}_{human,\text{ATT}}$) to do/possess the ATTRIBUTE, and the affinity for the given TOPIC ($\mathcal{A}_{\text{TOPIC,ATT}}$) to do/possess the ATTRIBUTE. We use COMET's ConceptNet relations to compute these affinities; specifically, we use the *CapableOf* relation. To approximate $\mathcal{A}_{human,\text{ATT}}$, we compute the average *CapableOf* score between the given ATTRIBUTE and all words in our previously defined HUMANSET. To compute $\mathcal{A}_{\text{TOPIC,ATT}}$, we compute the *CapableOf* score between the TOPIC and its ATTRIBUTE. The final animacy score of a TOPIC-ATTRIBUTE pair is defined as the difference $\mathcal{A}_{human,\text{ATT}} - \mathcal{A}_{\text{TOPIC,ATT}}$. If there are multiple TOPIC-ATTRIBUTE pairs, we consider the average animacy of all pairs.
2. Fluency – The de-personified sentences should be grammatically correct and sound natural. To measure for fluency, we use BART's generation scores (i.e. sum of individual token logits in the generated output).
3. Meaning Preservation – It is important that the de-personified sentence does not stray too far from the meaning of the original personification. We use BERTScore [192] between the de-personified and original sentences to measure meaning preservation.

We design a composite scoring metric comprised of the aggregate scores from these 3 metrics. Due to scaling differences, we consider the $\log$ of the animacy score. To account for the fact that lower animacy scores imply less animate TOPIC-ATTRIBUTE pairs (which is desirable in de-personification), we take the negative of the animacy. More precisely, we define our candidate

score $S_i$ for candidate $i$ as

$$S_i = \alpha \cdot (-\log(S_{anim.})) + \beta \cdot S_{flue.} + \gamma \cdot S_{mean.}$$

where $\alpha, \beta, \gamma$ are parameters. [5]

Once $S_i$ is computed for all candidates, we select the candidate with the highest composite score as our final de-personified sentence. A diagram of the entire **PINEAPPLE** pipeline is shown in Figure 3.3, and example outputs can be found in Table 3.1.

### 3.2.3 Test Data Construction

While automatically generated pairs of personifications and literal de-personifications may greatly assist with training, these may not necessarily be accurate for testing. Rather, it would be more ideal during testing if we have ground-truth human-annotated data. To mimic our task at hand, we gather a list of non-personified English sentences.[6] We then select two annotators who are native English speakers currently enrolled in a university with English as a medium of instruction. These annotators were instructed to manually personify these sentences to create ground-truth reference personifications. The final *PersonifCorp* test split has 72 literal + personified sentence pairs.

## 3.3 Experimental Setup

### 3.3.1 Methods

Below we outline the three models we consider, with two of them being naive baselines (COMET and Baseline-BART) that we simply use on *PersonifCorp*'s test set, and the third (Finetuned-BART) being our formulated model trained on *PersonifCorp*.

1. **COMET**: We extract the TOPIC-ATTRIBUTE pairs and identify the inanimate TOPICS using the methods detailed in §3.2.2. Instead of generating candidate replacements using BART like in §3.2.2, we generate candidates by considering the top $k = 10$ results for a given TOPIC using COMET's ConceptNet *IsCapable* relation (if the original ATTRIBUTE is a verb) or *HasProperty* relation (if adjective or adverb). To incorporate a notion of animacy,

[5] We use $\alpha = 1, \beta = 1, \gamma = 1$. Details about the tuning and selection of $\alpha, \beta, \gamma$ can be found in Appendix 3.7.3.

[6] https://github.com/tuhinjubcse/SimileGeneration-EMNLP2020#set-up-data-processing-for-simile

we use the previously defined ATTRIBUTE animacy $\mathcal{A}_{human,\text{ATT}}$ and select the candidate with highest animacy as our final replacement.

2. **Baseline-BART (BL-BART)**: We imitate the process outlined for the COMET baseline, except we use a pretrained BART model to generate the candidates instead of using COMET. All other steps (TOPIC-ATTRIBUTE extraction and candidate selection) remain the same.
3. **PINEAPPLE-BART (PA-BART)**: We fine-tune a BART model by supplying the *PersonifCorp* train split literal de-personified sentences (from the **PINEAPPLE** pipeline) as inputs, and the original ground-truth personifications as target outputs. This is trained as a seq2seq task. During generation, we use beam search. Further details are outlined in §3.3.3.

### 3.3.2 Evaluation

We consider both automatic evaluation metrics (§3.3.2) and human evaluation (§3.3.2).

#### Automatic Evaluation

For each model in §3.3.1, we evaluate its generated outputs on *PersonifCorp*'s test split using each of the following automatic evaluation metrics:

1. **BLEU** [125]: We use BLEU to ensure that the generations do not greatly differ from the inputs. We compute the BLEU score of each generated output with the literal inputs (for meaning preservation), as well as the ground-truth reference personifications.
2. **BERTScore** [191]: BERTScore measures how semantically related two sentences are, and is generally more robust than BLEU. We compute the BERTScore of each generated output with the inputs, as well as the ground-truth reference personifications.
3. **Fluency**: To approximately measure the fluency of a sentence, we use generation (log-perplexity) losses of each output using the GPT-2 language model [136].
4. **Animacy**: We are interested in how *personified* the generated output is. We use the same animacy metric used for candidate selection in §3.2, which is a combination of how animate the ATTRIBUTE is, as well as how inanimate the TOPIC is. More precisely, this is defined as $\mathcal{A}_{human,\text{ATT}} - \mathcal{A}_{\text{TOPIC,ATT}}$, where the $\mathcal{A}$ animacy scores are previously defined in §3.2.

**Human Evaluation**

The human evaluation was conducted using paid annotators on Amazon Mechanical Turk (AMT). Annotators were from Anglophone countries with $> 97\%$ approval rate.[7] Each test example was evaluated by exactly 2 annotators. For each test example, we first generate outputs using each of the methods outlined in §3.3.1. Corresponding to this test instance, we then create an AMT task page (a HIT), presenting the input literal sentence and each of the method outputs (in randomized order) for annotation along five aspects of text quality.

Specifically, annotation was elicited for the following metrics: **(1) Personificationhood** (*"To what extent does the new sentence contain a personification?"*), **(2) Appropriateness** (*"Do the personified nouns, verbs, adjectives, adverbs sound mutually coherent and natural?"*), **(3) Fluency** (*"Does it sound like good English with good grammar?"*), **(4) Interestingness** (*"How interesting and creative a rephrasing of the original sentence is the personified sentence?"*), and **(5) Meaning Preservation** (*"Do the entities, their actions, interactions, and the events appear and relate to each other in the same way as in the original sentence?"*). Each metric was scored on a Likert scale, with 1 being the lowest and 5 being the highest.

For *Interestingness*, we observed poor agreement scores amongst the AMT annotators.[8] Hence, for this aspect, we instead used a curated group of known, in-person annotators: a cohort of three native English-speaking students from an American university. Amongst these annotators, we observe a considerably higher agreement, with a Krippendorff $\alpha$ value of $0.5897$. For selecting this cohort from a slightly larger pool of candidates, we assessed their performance on a short qualification test of basic English literary skills and knowhow. The final cohort chosen each scored $85\%$ or higher on this test. Further details are in Appendix 3.8.3.

### 3.3.3 Implementation Details

The *PersonifCorp* training corpus was randomly split into a training and validation split with an 80-20 ratio. We fine-tune a BART-base model with 139M parameters using a learning rate of 2e-5 and a batch size of 4. Training was done for 20 epochs and 400 warmup steps, and model/epoch selection was performed based on the lowest validation loss. For generating the outputs, decoding was done using beam search with a beam size of 10. Additional details can be found in Appendix 3.9.

[7] More details about the human eval are in Appendix 3.8.1.
[8] Further details on inter-annotator agreement scores can be found in Appendix 3.8.2.

| | BLEU | | BERTScore | | | |
|---|---|---|---|---|---|---|
| | **Input** | **Gold** | **Input** | **Gold** | **Fluency ↓** | **Animacy** |
| Human Annotation | 0.172 | 1.000 | 0.749 | 1.000 | 5.264 | 0.332 |
| COMET | 0.127 | 0.128 | 0.655 | 0.569 | 6.366 | -2.028 |
| BL-BART | 0.132 | 0.133 | 0.728 | 0.617 | **4.573** | 0.106 |
| PA-BART | **0.153** | **0.160** | **0.748** | **0.636** | 5.460 | **0.679** |

Table 3.2: Average automatic evaluation results. The best-scoring method for each metric is highlighted in **bold**. Higher scores are better for all metrics except for fluency.

| | **Personificationhood** | **Appropriateness** | **Fluency** | **Interestingness** | **Meaning Preservation** |
|---|---|---|---|---|---|
| Human Annotation | 3.763 | 4.175 | 4.138 | 3.667 | 3.913 |
| COMET | 3.525 | 3.563 | 3.738 | 1.801 | 3.550 |
| BL-BART | 3.500 | 3.938 | **4.188** | 2.006 | 3.750 |
| PA-BART | **3.738** | **4.000** | 4.138 | **2.782** | **3.875** |

Table 3.3: Average human evaluation results. The best-scoring method for each metric is highlighted in **bold**.

## 3.4 Results and Analysis

### 3.4.1 Automatic Evaluation Results

Table 3.2 reports the automatic evaluation results for each of the metrics detailed in §3.3.2. We observe that our PA-BART model performs best across all automatic metrics except for fluency, where BL-BART performs best. The difference in performance is most significant in the *Animacy* metric, which is the key metric that quantifies the degree to which a sentence is personified. This confirms that indeed, our formulated **PINEAPPLE** method is successful in training a model to personify text.

Our PA-BART model also performs well for both BLEU and BERTScore, scoring better than the COMET and BART baselines, and coming second only to the human-written personifications.

Lastly, with regards to fluency, the BL-BART model outperforms the PA-BART model. This is likely because when considering GPT-2 likelihood, it may unfavorably penalize creative sentences with personifications since these are naturally less common in regular text. As an example, the sentence *"The stars danced playfully"* (GPT-2 loss = 7.02) would be deemed significantly less fluent than the sentence *"The stars twinkled brightly"* (GPT-2 loss = 5.24), even though they are both valid sentences with similar meanings. This argument is further supported by the fact that even the reference human-generated personifications received a lower fluency score than the BL-BART outputs. Further, literal sentences are indeed typically more *fluent* overall than personifications since they express the meaning literally. Nevertheless, we are still interested in the

other qualities being measured by fluency: *Is the sentence coherent? Does it make unnecessary grammatical errors?* In this regard, the fluency of PA-BART remains quite good. It is significantly better than the fluency of the COMET personifications and only slightly worse than the fluency of the human-written personifications.

### 3.4.2 Human Evaluation Results

Human evaluation results are reported in Table 3.3. Out of the five human evaluation metrics, the most pertinent metric to the personification generation task is *Personificationhood*, as this metric explicitly tries to quantify the presence and overall quality of personifications. In this metric, our PA-BART model performs significantly better than both baselines and is only slightly worse than the human reference personifications. This indicates that PA-BART is very successful in generating personifications that humans are able to detect and understand.

Aside from measuring the presence of personifications, we also want to measure more fine-grained qualities of these personifications. This is done by considering the *Appropriateness* and *Interestingness* scores. In *Interestingness*, PA-BART significantly outperforms both baselines but is worse than human annotations, while in *Appropriateness*, PA-BART slightly outperforms BL-BART and is slightly worse than human annotations. Overall, we can conclude that the personifications generated by PA-BART are of good quallity: the ATTRIBUTES match up well with the TOPICS, and they are overall very creative. This is further exemplified through the qualitative examples explored in §3.4.3.

Observations from *Meaning Preservation* and *Fluency* are very similar to those from the BLEU/BERTScore/Fluency metrics in the automatic evaluations. For *Meaning Preservation*, PA-BART performs best among all models, and only slightly trails human references. Meanwhile, for fluency, BL-BART was ranked the most fluent, outperforming both PA-BART and the human references. As discussed previously, this is likely due to the fact that literal sentences are generally perceived to be more fluent than personifications.

### 3.4.3 Qualitative Analysis

Table 3.4 contains a list of color-coded qualitative examples for each method. In Figure 3.2, we previously outlined three main types of personification TOPIC-ATTRIBUTE pairs, namely the cases where ATTRIBUTE is a noun, a verb, and an adjective. The first three examples in Table 3.4 demonstrate the capacity of our PA-BART model to capture all three cases. In the first example, the literal verb in "*your phone* ***rings*** *out loud*" is replaced with the more appropriate animate verb

| Method | Text |
|---|---|
| Literal Input | You are at a business dinner with your boss when your phone rings out loud (ex.1) |
| Human Ref | You are at a business dinner with your boss when your phone starts singing out loud |
| COMET | You are at a business dinner with your boss when your phone beep out loud |
| BL-BART | You are at a business dinner with your boss when your phone rings and you answer out loud |
| PA-BART | You are at a business dinner with your boss when your phone yells out loud |
| Literal Input | In most horror settings, silence is key. (ex.2) |
| Human Ref | In most horror settings, silence is the protagonist. |
| COMET | In most horror settings, silence scary. |
| BL-BART | In most horror settings, silence is preferred. |
| PA-BART | In most horror settings, silence is a ghost. |
| Literal Input | Her relationships with family and friends are very difficult (ex.3) |
| Human Ref | Her relationships with family and friends behave very unusually |
| COMET | Her relationships with family and friends serious |
| BL-BART | Her relationships with family and friends have always been strong. |
| PA-BART | Her relationships with family and friends are very lonely |
| Literal Input | Then there weren't any more parties as the house became silent (ex.4) |
| Human Ref | Then there weren't any more parties as the house kept mum. |
| COMET | Then there weren't any more parties as the house build. |
| BL-BART | Then there weren't any more parties as the house fell into disrepair. |
| PA-BART | Then there were no more parties as the house lamented. |
| Literal Input | It was a moonless nights, the air was still and the crickets were silent (ex.5) |
| Human Ref | It was a moonless nights, the air was asleep and the crickets were silent |
| COMET | It cold outside a moonless nights, the air cold outside still and the crickets noisy |
| BL-BART | It was a moonless nights, the air was still and the crickets were calling. |
| PA-BART | It was one of those moonless nights, the air was tired and the crickets were silent |
| Literal Input | The sound hit Frank loud enough to make your ear hurt (ex.6) |
| Human Ref | The sound slapped Frank loud enough to make your ear hurt |
| COMET | The sound echo Frank loud enough to make your ear sense sound |
| BL-BART | The sound of Frank Sinatra is loud enough to make your ear ring. |
| PA-BART | The sound clapped loud enough to make your ear cry |

Table 3.4: Qualitative examples for personification: literal input, human writing, COMET, BL-BART, and PA-BART. More can be found in Appendix 3.10.

in "*your phone **yells** out loud*." In the second, "*silence is **key***" is replaced with a noun in "*silence is **a ghost***", while in the third example, the literal adjective "*very **difficult***" is replaced with the animate adjective "*very **lonely***". These examples illustrate the generative flexibility of our model and its capacity to generate diverse outputs with different parts-of-speech.

We also observe that the outputs for PA-BART generally capture the meaning of the original text (and surrounding context) more accurately than the other baselines. In fact, the personifications greatly enhance the expressiveness of some of these sentences. In the first example, PA-BART replaces '*rings*' with '*yells*', while COMET replaces it with '*beeps*', and BL-BART leaves '*rings*' unchanged and just adds more details. Given the context of the sentence, we see that '*yells*' is more appropriate, expressive, and consistent with the context. A similar argument can be made for most of the other examples in the table: for the third example, PA-BART replaces

the literal *"very **difficult**"* with the much more animate and expressive *"very **lonely**"*, which is a suitable word to describe a relationship. In the fourth example, the BL-BART model is able to successfully capture the meaning of *"the house **became silent**"* with *"the house **fell into disrepair**"*. Although the meaning is correct, *"fell into disrepair"* is more literal and does not contain a personification. Compare this with the PA-BART's choice to replace *"the house **became silent**"* with *"the house **lamented**"*, which fits with the overall context (*"Then there were no more parties..."*), and also greatly enhances creativity by invoking the vivid image of lamentation. Meanwhile, in the fifth example, BL-BART personifies *"the crickets were **silent**"* with *"the crickets were **calling**"*. However, this shift completely changes the meaning, so it is a rather poor choice of personification. In contrast, PA-BART rewrites *"the air was **still**"* as *"the air was **tired**"*, which is a reasonable personification that is consistent with the imagery in the sentence (*"moonless nights"*, *"crickets were silent"*). Hence, we see that PA-BART can generate creative and meaningful personifications, while simultaneously staying true to the spirit of the sentence.

We also point out that our model is not limited to single-word substitutions. Rather, it considers a holistic view of the entire sentence and modifies key segments as necessary. This allows PA-BART to handle compound phrases well: consider, for instance, the one-to-many-word substitution of *'key'* $\longrightarrow$ *'a ghost'* (example 2), and the many-to-one-word substitution of *"became silent"* $\longrightarrow$ *"lamented"* (example 4). More importantly, PA-BART is also able to simultaneously generate personifications in two disjoint parts of the sentence, as seen in the last example: "*The sound **clapped** loud enough to make your ear **cry***." Here, there are two personifications in *"sound **hit**"* $\longrightarrow$ *"sound **clapped**"*, and *"ear **hurt**"* $\longrightarrow$ *"ear **cry**"*.

This last example also demonstrates the imperfection of our method. Although the model is able to generate two personifications, it loses a component of the original sentence because the recipient of the action (*'Frank'*) has disappeared. This same issue of meaning or information loss is present in example 2, where our model's output of *"silence is **a ghost**"*, while a personification, actually contradicts the original text *"silence is **key**"*. BL-BART's output of *"silence is **preferred**"*, while not a personification, correctly preserves the original meaning, as does the human reference of *"silence is **the protagonist**"*. This suggests that the model may still need some improvements with balancing creativity and semantic preservation. Other possible weaknesses are outlined in §3.6.

**Novelty and Diversity Analysis**

We randomly sample 30 examples from the PA-BART generations- and manually identify the parts of the sentences that were personified, as well as the animate ATTRIBUTES used to personify the TOPICS. Among the 30 examples, there were 27 unique ATTRIBUTES, and only 3 repeats. Additionally, there were 9 examples which generated completely new ATTRIBUTES that were never before seen in the training set, which demonstrates that the model is able to sufficiently capture the essence of a personification, rather than just blindly memorizing ATTRIBUTES from the training data.

## 3.5 Related Work

We present the linguistic underpinnings behind the TOPIC-ATTRIBUTE framework used in this chapter and explore how other types of figures of speech are generated. We also explore what makes personification generation so challenging.

**Linguistic Motivations.** Personifications traditionally do not have clearly defined classifications. In fact, even within the linguistic community, the definition of a personification is not always very clear-cut [37, 59]. A study by Long [105] examines the personification structure "*nonhuman subject + predicate verb (used for humans only) + others*," as well as the structure "*others + predicate verb (used for humans only) + nonhuman object + others*." We generalize and repackage these concepts, renaming the *subject* as the TOPIC and the *predicate verb* as the ATTRIBUTE. In doing so, we are able to capture more general notions of animacy beyond just verbs.

**Metaphor and Simile Generation.** A lot of studies on metaphors have focused on identification using techniques like word sense disambiguation [11], topic modeling [63, 167], dependency structures [73], and semantic analysis [68]. In terms of generation, early systems have explored grammar rules [53], while more recently, large language models have greatly aided in metaphor generation. Most notably, Stowe et al. [166] generate metaphors by considering conceptual mappings between certain domains and verbs. Chakrabarty et al. [19] further build on this by creating a parallel corpus of metaphors and training a large language model to perform the generation.

We also note here that the two aforementioned studies already cover personifications to a certain extent. However, these studies considered personifications as subtypes of metaphors. Some of the methods used may not generalize well to other types of personifications. Our study is the first to focus specifically on generating personifications.

For generating similes, Chakrabarty et al. [18] formulate using style-transfer models with COMET commonsense knowledge to generate similes. The study similarly creates a parallel corpus and trains a seq2seq model to perform the generation.

**Personifications.** There are currently few studies that specifically work on personifications. Gao et al. [50] detect personifications as a subtype of metaphors, but not as its own figure of speech. Generation is largely unexplored. We believe this is likely because personifications are generally more difficult to define and categorize. Furthermore, because several sources simplify personifications to fall under metaphors [19**?** ], there is also a lack of personification-specific datasets.

## 3.6 Conclusion

In this chapter, we studied the Constrained Creative setting of personification generation. To facilitate this study, we curated a dataset of personifications and formulated the **PINEAPPLE** method to automatically de-personify text.

The dataset we curated only had $\approx$ 350 examples of target-side, personified sentences. Thus, we had the poor Data Availability characteristic of this class of settings being present, not just in the number but also through the incomplete nature of parallel examples

We first hypothesize an underlying Creative Story based on the *elements of personification* (see §3.2.1 for more). Specifically, we posit a TOPIC-ATTRIBUTE relationship between the TOPIC, which is an inanimate entity and its animacy-requiring ATTRIBUTE, which is a dependent based on the dependency structure. This TOPIC-ATTRIBUTE structure is illustrated through examples in Figure 3.2.

Based on this creative story, we devised an Intervention to the typical E2EN2PP corpus curation process (see §3.2.2 for more) to impute source-side inputs. Specifically, this was accomplished through a "de-personification" pipeline to construct noisy source-side inputs $x_{noisy}$ which replace the animacy-requiring portions of the sentence with equivalent animacy-agnostic ones. using off-the-shelf dependency parsing, commonsense knowledge bases and pre-trained BART infilling followed by a 3-component candidate heuristic to enforce loss of ATTRIBUTE animacy, whilst preserving fluency and meaning

Using the imputed pseudo-parallel corpus thus constructed, *PersonifCorp*, we trained a seq2seq model (BART) to generate creative personifications. Through automatic, human, and qualitative evaluation, we demonstrated that these personifications make sentences more interesting and enhance the text's overall appeal. Our finetuned model successfully does this while maintaining a

high level of fluency and meaning perservation.

Some weaknesses of our model include failing to personify more complex sentence structures, as well as occasionally failing to preserve the exact meaning of the original sentence. We also believe that our model still has room to grow in terms of the diversity of personifications being generated. A promising future direction would be to explore possible ways to acquire more control over which parts of the sentence are being personified or what types of personifications are being generated.

## 3.7 Appendix A: De-Personification Pipeline

### 3.7.1 Dependency Tree Merging Example

Figure 3.4 contains an example of the merging process that was described in the TOPIC-ATTRIBUTE extraction step in §3.2. As outlined in §3.2, edge relations to iteratively merge are *negation modifiers*, *possession modifiers*, *nominal modifiers*, *adjectival complements*, and *objects of prepositions*, as well as words tagged as determiners and parts of compound phrases. The priority order for merging is as follows: 1) compound phrases, 2) nominal modifiers, 3) possession modifiers, 4) negation modifiers, 5) determiners, 6) objects of prepositions, 7) adjectival complements.

### 3.7.2 Human-Related Words

In §3.2, we defined the *IsAPerson* animacy metric as the average of the *IsA* scores between the TOPIC and various words that are very closely related to *'human'*. We called this set the HUMANSET. The words contained in HUMANSET are as follows: {"person", "human", "man", "woman", "human being", "boy", "girl"}.

These words were empirically selected by considering a list of synonyms of the word *'person'* and checking the *IsA* relation COMET scores with the word *'human'*. All of the above words have *IsA* scores with *'person'* of less than 5.10.

### 3.7.3 Parameters and Thresholds

***IsAPerson* Threshold.** For the *IsAPerson* animacy score, we use a threshold of 7.0. *IsAPerson* scores $< 7.0$ are considered animate, while scores $\geq 7.0$ are considered inanimate. This threshold was selected empirically using words known to be animate and words known to be inanimate. Words tested include "she" (5.31), "person" (6.41), "moon" (8.743), "opportunity" (9.488), "stars" (8.717), "joe" (5.804), "jane" (4.976), "the police officer" (6.462), "my friend" (6.805), "my new iphone" (10.055). From these observations, we observe that all animate words have an *IsAPerson* score of $< 7.0$, while all inanimate objects have a score of $\geq 7.0$. We hence conclude that $7.0$ is a suitable threshold.

**Candidate Selection Composite Score Parameters.** For the $\alpha, \beta, \gamma$ used in the composite score for candidate selection, we use values of $\alpha = 1, \beta = 1, \gamma = 1$. This was selected for two reasons. First, all of the score values had largely similar scales (logarithmic), so setting $\alpha, \beta, \gamma$ to a larger value like 2 or 3 would disproportionately favor a certain metric, which is not

| Metric | Spearman Correlation | Krippendorff $\alpha$ |
|---|---|---|
| Personificationhood | 0.0934 | 0.0250 |
| Appropriateness | 0.1660 | 0.1778 |
| Fluency | 0.0050 | 0.0942 |
| Interestingness | 0.6160 | 0.5898 |
| Meaning Preservation | 0.0389 | 0.2558 |

Table 3.5: Inter-annotator agreement scores.

what we desire. Second, we experimented with using values such as 0.8, 1.2, and 1.5, but the generated de-personifications were either very similar or slightly worse than the default setting of $\alpha = 1, \beta = 1, \gamma = 1$. A possible future direction would be to explore possible values of $\alpha, \beta, \gamma$ more thoroughly, but for this dataset, we stick to the simple case of $\alpha = 1, \beta = 1, \gamma = 1$.

## 3.8 Appendix C: Evaluation Details

### 3.8.1 Human Evaluation Setup

A total of 20 unique AMT annotators participated in the study for fluency, appropriateness, and meaning preservation, each performing 4.0 HITs on average. Annotators were compensated 1.12$ per HIT, each of which was designed to take <6 mins on average.

22 unique AMT annotators participated in the second, separate study for personificationhood, each performing 4.36 HITs on average. Annotators were compensated 0.56$ per HIT, each of which was designed to take <2 mins on average.

For the interestingness study, the details regarding annotator background and selection can be found in §3.3.2 and Appendix 3.8.3.

The html templates including instructions, questions and other study details corresponding to both these AMT studies can be found in the `templates/` subfolder of our code submission zip, with the names `fluency_appropriateness_meaningPreservation.html` and `personificationhood.html` respectively.

### 3.8.2 Inter-Annotator Agreement Scores

Each generated input instance and its respective model outputs are labelled by two distinct annotators. To measure inter-annotator agreement, we use Spearman correlation and Krippendorff $\alpha$, as reported in Table 3.5.

To get the Spearman correlation point value for a given aspect and test instance, we compute mean pairwise Spearman correlation between the aspect values assigned to the corresponding

model outputs by every pair of annotators. Specifically, we use the *scipy.stats* implementation to compute this. [9]

For Krippendorff $\alpha$, we treat each human evaluation aspect as an ordinal quantity. Specifically, we use the implementation provided by the python library *krippendorff 0.5.1*.[10]

### 3.8.3 English Assessment Test for Annotators

From the native English-speaking university student annotators who enrolled to participate in our Interestingness study, we first elicited answers to an English assessment test, as mentioned in §3.3.2.

The assessment test comprised of 12 questions spanning multiple question types testing the examinee's grasp of the use and distinction between various figures of speech, basic literary general knowledge, and verbal reasoning skills. A spreadsheet file containing this test can be found with the name *LiteratureTest.xlsx* under the *Templates/* subfolder of our code submission .zip file.

The final annotators used for our interestingness study were chosen from those who got 11 or more of the 12 questions on the English assessment test correct, hence scoring at least $85\%$ on the test.

## 3.9 Appendix B: Implementation Details

The BART-base model was trained using a learning rate of 2e-5. This was by conducting a hyperparameter search over the values {1e-6, 5e-6, 1e-5, 2e-5, 5e-5, 1e-4} and selecting the model/epoch based on lowest validation loss. The same process was done to select a batch size of 4 using a hyperparameter search over values {2,4,8,16}. Training was done for 20 epochs and 400 warmup steps. The Adam optimizer was used, and inputs were truncated to a maximum length of 64 tokens (using BART's subword tokenization).

Training was done on Google Colaboratory environments using V100 GPUs. For the BART-base model, a single training loop of 20 epochs takes approximately 10 minutes to complete.

[9] https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.spearmanr.html

[10] https://pypi.org/project/krippendorff/

| Method | Text |
|---|---|
| Literal Input | The news hit me hard. (ex.7) |
| Human Ref | The news punched me hard. |
| COMET | The news report event late me hard. |
| BL-BART | The news hit me hard. |
| PA-BART | The news pounded me hard. |
| Literal Input | Panic is sweeping through the streets contagiously. (ex.8) |
| Human Ref | Panic is driving around the streets contagiously. |
| COMET | Panic is result through the streets contagiously. |
| BL-BART | Panic is running through the streets contagiously. |
| PA-BART | Panic runs through the streets contagiously. |
| Literal Input | In the dead of summer, in Ohio, the clothes felt hot and humid. (ex.9) |
| Human Ref | In the dead of summer, in Ohio, the clothes seemed to be sweating themselves. |
| COMET | In the dead of summer, in Ohio, the clothes fit and humid. |
| BL-BART | In the dead of summer, in Ohio, the clothes we wear can be hot and humid. |
| PA-BART | In the dead of summer, in Ohio, the clothes complained and humid. |
| Literal Input | Streaks of electricity zipped through the air, swirling around and around dangerous (ex.10) |
| Human Ref | Streaks of electricity cackled through the air, swirling around and around dangerous. |
| COMET | Streaks of electricity stain carpet through the air, swirling around and around dangerous. |
| BL-BART | Streaks of electricity were shooting through the air, swirling around and around the building. |
| PA-BART | Streaks of electricity danced around and around dangerous. |
| Literal Input | The sun is nowhere to be found, and the streets are eerily quiet (ex.11) |
| Human Ref | The sun seems to be on leave, and the streets are eerily quiet. |
| COMET | The sun down, and the streets full of traffic. |
| BL-BART | The sun, and the streets. |
| PA-BART | The sun refuses to be found, and the streets are polite |
| Literal Input | The wind blew through me fast. (ex.12) |
| Human Ref | The wind stampeded through me fast. |
| COMET | The wind howl through me fast. |
| BL-BART | The wind was going through me fast. |
| PA-BART | The wind ran me fast. |

Table 3.6: Additional qualitative examples for personification outputs: literal input, human writing, COMET, BL-BART, and PA-BART.

## 3.10 Appendix D: Additional Examples

Table 3.6 is an extension of Table 3.4 and contains additional qualitative examples of the generations.

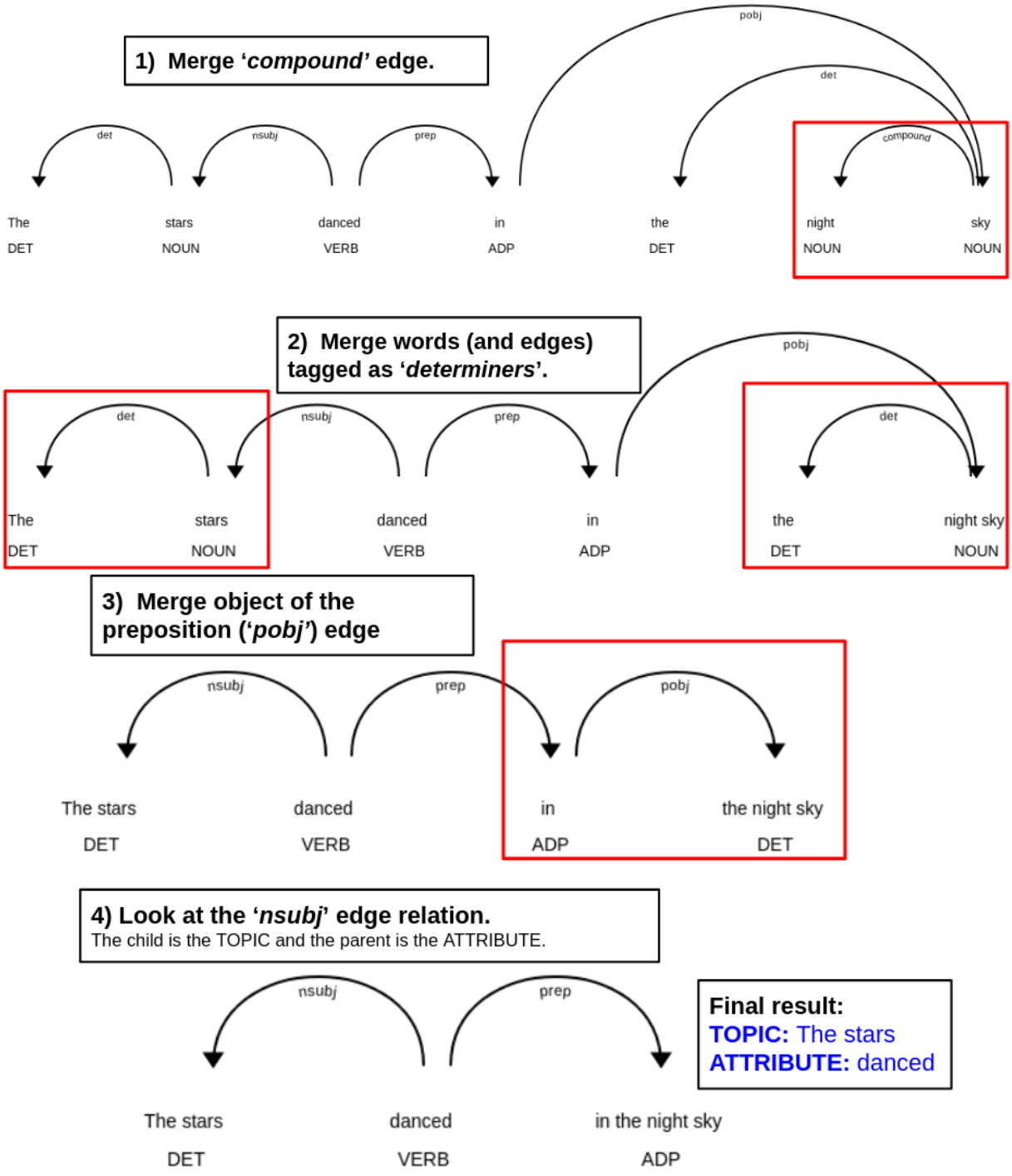



Figure 3.4: Step-by-step example of the merging process for TOPIC-ATTRIBUTE identification.

# Chapter 4

# Tongue Twister Generation

## (Under Review @ EMNLP 2022)

> I have a sift of sifted thistles and unsifted thistles because I'm a thistle sifter.
>
> George VI — *The King's Speech*

A *tongue twister* is a sentence that is both *articulatorily difficult* (i.e. colloquially speaking, *hard to say* or "twisting". The process of automatically generating tongue twisters is challenging since the generated utterance must satisfy two conditions at once: phonetic difficulty and semantic meaning. Furthermore, phonetic difficulty is itself hard to characterize and is expressed in natural tongue twisters through a heterogeneous mix of phenomena such as alliteration and homophony. In this chapter, we devise a set of related approaches named **PANCETTA**: **P**honeme **A**ware **N**eural **C**ompletion to **E**licit **T**ongue **T**wisters **A**utomatically. We leverage phoneme representations to capture the notion of phonetic difficulty, and we train language models to generate original tongue twisters on two proposed sub-settings. To do this, we curate a dataset called *TT-Corp*, consisting of existing English tongue twisters. Through automatic and human evaluation, as well as qualitative analysis, we show that **PANCETTA** generates novel, phonetically difficult, fluent, and semantically meaningful tongue twisters.

In this chapter, we study the **Constrained Creative** NLG setting of tongue twister generation from prompts. Specifically, we study two sub-settings — the first with textual prompts and the second with keywords as prompt.

The **CG Definition/Constraints** for this setting are: Given a short textual or keyword prompt, generate a completing sequence of graphemes such that it:

1. Forms a fluent, meaningful sentence
2. Is also articulatorily difficult i.e., forms a sequence of phonemes that is “hard to say".

We outline and describe in detail our **Data Availability** scenario in §4.2. The coining of a novel, unique tongue twister that spreads sufficiently to become normative and well-recognized is rare, hence making them a long-tailed linguistic phenomenon [117]. As training data, we only have access to $\approx 644$ examples of tongue twisters including their accompanying prompts.

We posit an underlying **Creative Story** for our setting in §4.1.1 , based on the intuition of a “*mouth model*". Based on this intuition, we devise a two-step **Intervention** to the typical E2EN2PP training process, as detailed in §4.3.1 and illustrated in Figure 4.1.

1. At training time, instead of finetuning GPT-2 in grapheme-to-grapheme (G2G) mode alone, as would be typical, we finetune it to generate either of phoneme/grapheme completions from prompts in either of phoneme/grapheme form.
2. At inference time, use grapheme-to-grapheme (G2G) mode to infer from learnt model. The other 3 finetuning modes (G2P, P2G,G2G) merely served as a training-time, “scaffolding" mechanism.

## 4.1 Introduction

A *tongue twister* is a sentence that is both *articulatorily difficult* (i.e. colloquially speaking, *hard to say* or "twisting" while at the same time being *meaningful and fluent*. Some examples of tongue twisters are shown in Table 4.1.

Together with riddles, rhymes, fables, and other such creative artifacts, tongue twisters were historically often employed as a vehicle for early transmission of native language diction, grammar, and vocabulary to children, through parent-child interaction, playtime activity, and kindergarten instruction [1, 111]. Tongue twisters have also been used as experimental aides for research studies of speech production in cognitive science and related disciplines, both amongst healthy speakers and those with speech and auditory disorders such as dysarthia [81]. They are also used as pedagogic aids in speech therapy and treatment of speech disorders, and psychological disorders relating to public speaking and elocution [148]. An example of this was in a scene[1] from *The King’s Speech (2010)*, where George VI repeats a tongue twister during therapy to reduce his stutter. Lastly, they find use as teaching aides for English diction in EFL (English as a Foreign Language) instructional settings [135].

[1] https://youtu.be/7WJts0gKCRM?t=53

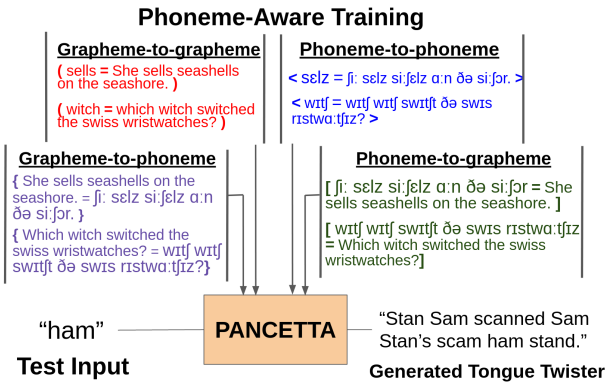



Figure 4.1: Overview of the phoneme-aware training in the **PANCETTA** model.

| Task | Input | Tongue Twister | Tongue Twister (Phoneme) |
|---|---|---|---|
| TT-Prompt | A good cook | A good cook could cook as many cookies as a good cook who could cook cookies. | ə gʊd kʊk kʊd kʊk æz mɛni kʊkiːz æz ə gʊd kʊk huː kʊd kʊk kʊkiːz. |
| | Chubby jugglers | Chubby jugglers juggling oranges jovially. | tʃʌbi dʒʌgəlɾz dʒʌgəlɪŋ ɔrəndʒəz dʒoʊveɪli. |
| | Does the | Does the rapid rabid rabbit wrap it? | dʌz ðə ræpəd ræbɪd ræbət ræp ɪt? |
| TT-Keyword | shoes, dog | If a dog chews shoes, whose shoes does he choose? | ɪf ə dɔg tʃuːz ʃuːz, huːz ʃuːz dʌz hiː tʃuːz? |
| | blood, death | Bad dead bed-bugs bleed bug blood. | bæd dɛd bɛd-bʌgz bliːd bʌg blʌd. |
| | king, art, wall | A truly rural frugal ruler’s mural was on the wall. | ə truːli rʊrəl fruːgəl ruːlɾz mjʊrəl wɑːz ɑːn ðə wɔl. |

Table 4.1: Example inputs and target outputs for both the TT-Prompt and TT-Keyword sub-settings, along with the phoneme representations of the tongue twisters.

The coining of a novel, unique tongue twister that spreads sufficiently to become normative and well-recognized is rare, hence making them a long-tailed linguistic phenomena [117]. However, they are not limited to English, and are found across the world’s languages, e.g., Persian (*"Shish sikh jigar sikhi shi shezar."*) [72] and French (*"Cinq chiens chassent six chats."*)

### 4.1.1 The Mouth Model

Consider the idealized scenario where we have a *"mouth model"* that a) maps different regions of the mouth, palate, and the larynx to the dictionary of fundamental sounds, i.e. phones being produced, and b) based on this grounding, can quantify the hardness of producing one sound

after another, i.e. induces a distance measure between any phone pair. Assuming access to this idealized model, one could deconstruct the process of generating a tongue twister as sampling a sequence of preferably difficult (distant) phone-phone transitions starting with an initial sequence of one or more phones (that could come from a prompt, or be chosen uniformly, based on the sub-setting).

However, there are several impediments that make realizing such an idealized model intractable. Firstly, the dictionary of fundamental sounds at the granularity we use in practice, i.e. at the level of phonemes, does not neatly map to particular points of the palate [92]. Rather, each phoneme itself corresponds to an action(s) involving multiple organs and palatal regions e.g., velar consonants like *k* are produced based on tongue-velum (soft upper palate) interaction. Secondly, a tongue twister as per its definition is not merely a difficult to pronounce sequence of phonemes — but also one that maps to a meaningful and fluent sequence of words. How one can maintain this property in conjunction with the process of sampling difficult transitions from the mouth model's space is unclear.

### 4.1.2 Challenges & Contributions

The automatic generation of tongue twisters has largely been unexplored. This task is challenging because it requires being able to model phonetic difficulty of various syllables and tokens, that is not something that existing language models are trained to do. To achieve this, we have to work in the phoneme space. Phonemes have previously been used to aid in speech recognition [168] and rhyme generation [67]. We hypothesize that by working with phonemes, we will be able to model and generate patterns that characterize phonetic difficulty.

Tongue twisters go beyond relatively simpler phonetic phenomena such as alliteration, since they employ a heterogeneous mix of strategies based on such phenomena [77] including alliteration itself (***s**he **s**ells **s**eashells*), use of homophonic words/subwords (*sells*/*-shells*, *she*/*sea-*), and alternating between similar start phonemes for tokens (*s* and *sh*), sometimes even using these co-operatively within the same example to create the cumulative effect of articulatory difficulty.

Our contributions are as follows: (1) We curate a dataset, *TT-Corp*, of diverse tongue twisters. (2) We present two new sub-settings (TT-Prompt and TT-Keyword) for automatic tongue twister generation, and we design and evaluate simple baselines for these tasks. (3) We formulate a phoneme-aware method called **PANCETTA**, that models and generates coherent and phonetically difficult phrases by taking phonemes into account. We show that **PANCETTA** generates higher-quality tongue twisters through both automatic and human evaluations and qualitative

analysis of the outputs.

## 4.2 Sub-settings and Dataset

### 4.2.1 Sub-settings

We formulate two sub-settings for automatic tongue twister generation. We call these tasks TT-Prompt and TT-Keyword, and they are detailed below:

1. **Generating tongue twisters from prompts (TT-Prompt)**: Given a few words to start a sentence, the goal is to complete the sentence in a coherent way such that the resulting generation is a tongue twister. Prompts can be of varying lengths. Examples of this task can be found in Table 4.1.

2. **Generating tongue twisters from keywords (TT-Keyword)**: Given a set of keywords, the goal of this task is to generate a coherent tongue twister that incorporates the semantics of the keywords. The set of keywords can be of varying sizes. These keywords do not necessarily have to appear verbatim and do not necessarily have to appear in order. Examples of this task can be found in Table 4.1.

### 4.2.2 *TT-Corp* Dataset

We curate a dataset of 644 unique English tongue twisters into a dataset called *TT-Corp*. These tongue twisters are compiled from various sources, ranging from blog posts to English learning websites. A more detailed list of these sources and data processing details can be found in Appendix 4.8.

We also create a non-tongue twister version of each input in *TT-Corp*, that will later be used to explore style transfer models (§4.3.1) and to train a phonetic difficulty classifier (§4.3.2). This is done through synonym replacement. First, we determine the parts-of-speech of all the words in the sentence and identify the nouns, verbs, and adjectives.[2] We then use WordNet [41] to generate a list of synonyms for each of these nouns, verbs, and adjectives, and we select the highest ranked replacement that shares the same part-of-speech. Examples of this process are shown in Table 4.2.

One key advantage of this synonym replacement process is that it can replace a word according to its part-of-speech and function in the sentence. In the second example in Table 4.2, the word "desert" appears twice — first as a verb (which is replaced with "abandon"), and again as a

[2]The spaCy library was used to extract the POS tags.

| Tongue Twister | Non-TT Version |
|---|---|
| There was a little witch which switched from Chichester to Ipswich. (ex.1) | There was a small enchantress which exchanged from Chichester to Ipswich. |
| He wanted to desert his dessert in the desert. (ex.2) | He desired to abandon his sweet in the desert. |
| Tie a tight knot in the shape of a nought. (ex.3) | Bind a taut gnarl in the form of a zero. |

Table 4.2: Examples of the synonym replacement process to generate non-tongue twister versions of the tongue twisters in *TT-Corp*. Different colors are used to indicate which words are replaced by their synonyms.

noun (which is not replaced). However, this synonym replacement process does not take the context of the words into account. In the third example in Table 4.2, while the individual synonym replacements make sense on their own, the final sentence sounds quite unnatural. For our purposes, however, this is not a significant issue: we do not need the replacement sentences to be absolutely perfect, as the quality of the ground-truth tongue twister is more important. This will be explained further when we use this parallel dataset in §4.3.2.

## 4.3 Methodology

As this is a new task, there are no existing methods that can easily generate novel tongue twisters. The main challenge is how to incorporate phonetic difficulty into our generations. To do so, we formulate two baseline and two phoneme-aware models, that are applicable to both the TT-Prompt and TT-Keyword sub-settings (see Table 4.3).

### 4.3.1 Models

1. **Grapheme-based Methods (g2g)** — We treat tongue twister generation as a seq2seq task, where the prompt (for TT-Prompt) or keywords (for TT-Keyword) is the input, and the tongue twister is the target output. The intuition is that by fine-tuning using these tongue twisters, the model can implicitly acquire a notion of phonetic difficulty. We fine-tune GPT-2 and GPT-J using the inputs "([X]=[Y])", where [X] represents the prompt/keywords, and [Y] represents the tongue twister.

2. **Style Transfer Methods** — Given a prompt or a set of keywords, we first use GPT-2 to generate a sentence which is not necessarily a tongue twister. This is easy for prompts since GPT-2 is trained to do causal LM. For TT-Keyword, since GPT-2 is not trained to generate a

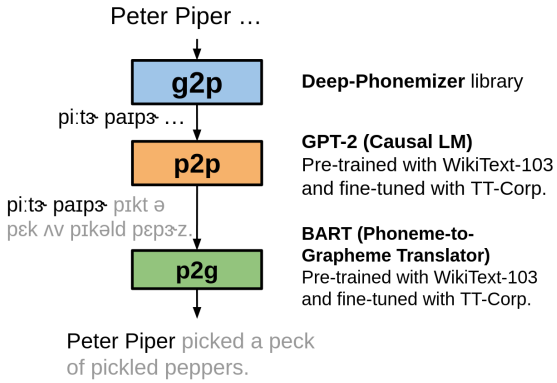



Figure 4.2: Overview of **PANCETTA-P** pipeline.

sentence from keywords, we first train a GPT-2 model for this task. We sample 10,000 sentences from WikiText-103 [113] and extract their keywords using KeyBERT [56]. We then fine-tune GPT-2 using an "([X]=[Y])" template as described in the g2g methods. Here, [X] represents the keywords, and [Y] represents the WikiText sentence.

We then attempt to convert these generated natural sentences into tongue twisters. We treat this as a seq2seq task and train a seq2seq model using our parallel dataset. During training, we use the non-TT versions as inputs and the tongue twisters as the ground truth target outputs. We use BART and T5 models for this seq2seq task.

3. **PANCETTA-P** (Phoneme) — For the previous g2g models, the fine-tuning was done only using graphemes. Because graphemes are not always representative of pronunciation, we hypothesize that it may be difficult for such models to capture information regarding the pronunciation. If we instead had a generative model that works in the phoneme space, then we could fine-tune this model on the tongue twister phonemes and hope that it can better capture these phonetic cues.

We first pretrain a GPT-2 model to perform causal LM generation for phonemes. Pretraining is done using the WikiText datasest: we first convert all the WikiText sentences into their IPA phoneme representations and train a GPT-2 model on it.[3] For TT-Keyword, instead of training a causal LM on regular phoneme sentences, we train to generate from keywords, using the ([X]=[Y]) template previously described. While there are multiple g2p phonemization toolkits, there are no readily available p2g toolkits that work well. Hence, we train our own p2g model. We treat this as a seq2seq translation task, once again using WikiText. We train a BART model with the phonemes as the inputs and the graphemes as the target outputs. Once both the p2p generation and the p2g translation models are trained, we then fine-tune the p2p models on the tongue twister

[3]The `deep-phonemizer` Python package was used for g2p transliteration.

phonemes (all steps similar to g2g), then use the p2g model to retrieve the grapheme representation of the outputs. Lastly, since there is no capitalization in the phoneme space, we have to fix to capitalization of the generated outputs. We use the FastPunct library for this.[4] (Note: unlike the previous g2g methods, we only use GPT-2 (and not GPT-J) for **PANCETTA-P** because GPT-J is too large to pre-train in a reasonable fashion.)

4. **PANCETTA-J** (Joint) — One drawback of **PANCETTA-P** is that because we do our own pre-training on the phoneme space, we are not able to leverage the existing pre-training of large language models. In order to leverage both the phoneme representations and the pre-training of GPT, we formulate **PANCETTA-J**. This is similar to g2g, but instead of only training with the template ([X]=[Y]), we train with 4 different templates, representing 4 different modalities. Specifically, we train with the templates ([PK_G]=[TT_G]), <[PK_P]=[TT_P]>, [[TT_G]=[TT_P]], and {[TT_P]=[TT_G]}, where PK represents the prompts/keywords, TT represents the tongue twisters, G represents the grapheme representation, and P represents the phoneme representation. These 4 modalities represent g2g, p2p, g2p, and g2g respectively. Here, the surrounding brackets function similar to separator tokens that indicate the modality for the model.

At test time, the model can be directly decoded using g2g mode without requiring phoneme information. We hypothesize that the phonetic structures learned during training time can serve as an effective "scaffold" — being used explicitly only during finetuning time [170].

### 4.3.2 Evaluation Metrics

#### Automatic Evaluation

As described in §4.1, a good tongue twister needs to satisfy two notions: it needs to be both difficult to pronounce, as well as semantically coherent. To evaluate the "tongue twisterness" of our generations, we consider these two notions separately.

1. **Phonetic Difficulty**: We fine-tune a pretrained BERT-base classifier to differentiate between tongue twisters and regular sentences. To train this model, we use the parallel dataset of tongue twister and non-TT pairs as described in §4.2.2. We specifically use these tongue twister and non-TT pairs so that the model specifically learns to classify based on phonetic difficulty rather than things such as semantics. However, as mentioned in §4.2.2, the replacement sentences may sometimes sound unnatural. To ensure that the classifier learns to differentiate tongue twisters

[4] https://github.com/notAI-tech/fastPunct

| Method Name | Models Used | Description | Phoneme Representation | Leverage Pretraining |
|---|---|---|---|---|
| Grapheme-based Methods | GPT-2, GPT-J | g2g | ✗ | ✓ |
| Style Transfer Methods | BART, T5 | g2g + g2g | ✗ | ✓ |
| **PANCETTA-P** | GPT-2, BART | g2p + p2p + p2g | ✓ | ✗ |
| **PANCETTA-J** | GPT-2, GPT-J | g2g, p2p, g2p, p2g (only g2g during test-time) | ✓ | ✓ |

| Method Name | Example |
|---|---|
| Grapheme-based Methods | She sells → She sells seashells on the seashore. |
| Style Transfer Methods | She sells → She sells things on the beach. → She sells seashells on the seashore. |
| **PANCETTA-P** | She sells → ʃiː sɛlz → ʃiː sɛlz siːʃɛlz ɑːn ðə siːʃɔr → She sells seashells on the seashore. |
| **PANCETTA-J** | She sells → She sells seashells on the seashore.<br>She sells seashells on the seashore. → ʃiː sɛlz siːʃɛlz ɑːn ðə siːʃɔr.<br>ʃiː sɛlz siːʃɛlz ɑːn ðə siːʃɔr. → She sells seashells on the seashore.<br>ʃiː sɛlz → ʃiː sɛlz siːʃɛlz ɑːn ðə siːʃɔr. |

Table 4.3: Summary of the models discussed in §4.3.1, along with some examples.

instead of picking up on these false signals, we augment our dataset with additional negative (non-TT) examples consisting of 500 sentences randomly sampled from WikiText. Rather than directly training on the sentences, we first convert the sentences to a phoneme representation and train a classifier on the phonemes.

2. **Fluency:** We not only want phonetically difficult sentences; they must also be fluent and coherent. To measure this, we use the generation (log-perplexity) losses from a pretrained GPT-2.

3. **Keyword Relevance** (only for TT-Keyword): In TT-Keyword, we want to ensure that the generated tongue twister is semantically similar to the keywords used. To measure this, we use the BERT embedding of keywords and compare it with the embedding of the target sentence. More specifically, we use the BERTScore [192] between the generated sentence and the "sentence" consisting of the keywords separated by commas.

We do not use metrics based on reference matching with the reference/gold outputs. This is because tongue twister generation from prompts/keywords is inherently a creative task with high output diversity and a large subspace of valid outputs.

### Human Evaluation

The human evaluation metrics are very similar to the ones in §4.3.2. These are as follows: (1) **Phonetic Difficulty** ("*How hard is the sentence to pronounce? To get a better sense of the difficulty, try saying the sentence out loud, quickly, and multiple times.*") and (2) **Fluency** ("*Does*

| Method | TT-Prompt | | TT-Keyword | | |
|---|---|---|---|---|---|
| | Phon. Difficulty | Fluency ↓ | Phon. Difficulty | Fluency ↓ | Keyword Relevance |
| g2g (GPT-2) | 0.774 | 5.433 | 0.786 | 5.224 | 0.795 |
| g2g (GPT-J) | 0.848 | 5.643 | 0.856 | 5.593 | 0.794 |
| Style Transfer (GPT-2+BART) | 0.672 | 4.356 | 0.472 | **3.662** | 0.783 |
| Style Transfer (GPT-2+T5) | 0.631 | **4.256** | 0.414 | 4.309 | 0.780 |
| **PANCETTA-P** (GPT-2+BART) | 0.794 | 5.986 | 0.871 | 6.596 | 0.801 |
| **PANCETTA-J** (GPT-2) | 0.785 | 5.244 | 0.803 | 5.058 | **0.803** |
| **PANCETTA-J** (GPT-J) | **0.866** | 5.718 | **0.888** | 5.169 | 0.800 |
| Gold Outputs | 0.925 | 5.745 | 0.925 | 5.745 | 0.812 |

Table 4.4: Automatic evaluation averages for both TT-Prompt and TT-Keyword. The best-scoring method for each metric is highlighted in **bold**. Higher scores are better for all metrics except for fluency.

*it sound like good English with good grammar?*"). Further details about the evaluation process can be found in §4.4.2.

## 4.4 Experimental Setup

### 4.4.1 Implementation Settings

**Prompt / Keyword Extraction:** To extract prompts for TT-Prompt, we simply consider the first three words of each sentence by checking for the whitespace character. To extract keywords for TT-Keyword, we use the KeyBERT library [56], that returns keywords ranked by their cosine similarity scores to the entire sentence itself. For each sentence, we consider the top 5 keywords as our set of keywords. Not all sentences have 5 keywords; some may have less. In these cases, we just take all the keywords.

**Dataset splits:** We split *TT-Corp* into a training-validation-test split with a 70-15-15 ratio. We use the same splits across all models and across both TT-Prompt and TT-Keyword sub-settings, as well as for training the phonetic difficulty classifier.

**GPT-2 fine-tuning** (g2g, **PANCETTA-P**, **PANCETTA-J**): We use the pre-trained GPT2-base (124M params.) and fine-tune for 5 epochs with a learning rate of 5e-5 and 100 warmup steps.

**BART pre-training** ( **PANCETTA-P**): Here, we train the p2g model. We pre-train BART-base (139M params.) on WikiText-103 phonemes. We split WikiText into train-validation-test splits of 80-10-10. The final training set has size 523k. Training was done for 20 epochs with batch size of 16, an initial learning rate of 5e-4, and a weight decay of 0.1 with a cosine scheduler.

**BART & T5 fine-tuning** (Style Transfer): We fine-tune BART-large (406M params.) &

T5-large (737M params.) for 30 epochs with a batch size of 16, learning rate of 2e-5, and 400 warmup steps.

**GPT-J fine-tuning** (g2g, **PANCETTA-J**): Because GPT-J is too large (6B parameters) and occupies too much memory, we instead use a compressed version of GPT-J [5], that incorporates various techniques such as 8-bit quantization [31] and low-rank adaptation [70]. To fine-tune, we use 10 epochs, a batch size of 1, and a learning rate of 1e-5.

For GPT-J and GPT-2, generation was done using nucleus sampling with p=1.0 and a temperature of 0.8. Meanwhile, for BART and T5, generation was done using beam search with a beam size of 5. More details about the hyperparameter search, runtime, and compute can be found in Appendix 4.10.

### 4.4.2 Human Evaluation Settings

Human evaluation was done on Amazon Mechanical Turk (AMT). We selected annotators with >97% HIT approval rate from Anglophone countries[6]. In each HIT, we present the generated outputs for each example in randomized order, and each test example was evaluated by exactly 2 annotators.

We conduct two rounds of annotation, one for TT-Prompt and another for TT-Keyword. Within each round, we further subdivide annotating GPT-2 experiments and GPT-J experiments. This is for two key reasons. The first reason is to ensure that we only have one independent variable and to ensure that the changes in performance are due to the methodologies rather than the size of the models. (This GPT-2/GPT-J split only applies to g2g and **PANCETTA-J** models; for the style-transfer and **PANCETTA-P** models, we keep the same models for both rounds of evaluation.) The second reason for subdividing GPT-2 and GPT-J experiments is so that we do not subject the annotators to information overload from having to annotate too many similar examples. Owing to the same consideration, we also decided to not perform human evaluation on the Style Transfer T5 baseline model because we found it very similar to Style Transfer BART.

| Method | TT-Prompt | | TT-Keyword | |
|---|---|---|---|---|
| | **Phonetic Difficulty** | **Fluency** | **Phonetic Difficulty** | **Fluency** |
| g2g (GPT-2) | 3.056 | 3.736 | 3.847 | **4.139** |
| Style Transfer (GPT-2+BART) | 2.569 | 3.639 | 3.500 | 3.819 |
| **PANCETTA-P** (GPT-2+BART) | **3.528** | **3.778** | 3.722 | 3.764 |
| **PANCETTA-J** (GPT-2) | 3.153 | 3.764 | **3.889** | 3.931 |
| Gold Outputs | 3.361 | 3.931 | 3.833 | 4.000 |
| g2g (GPT-J) | 3.521 | **3.979** | 3.791 | 3.708 |
| Style Transfer (GPT-2+BART) | 3.271 | 3.75 | 3.25 | 3.750 |
| **PANCETTA-P** (GPT-2+BART) | **3.854** | 3.708 | 3.833 | **3.896** |
| **PANCETTA-J** (GPT-J) | 3.708 | 3.646 | **3.979** | 3.604 |
| Gold Outputs | 3.750 | 4.000 | 4.104 | 3.729 |

Table 4.5: Human evaluation averages for TT-Prompt and TT-Keyword. Top method scores for each metric are **bold**.

## 4.5 Results and Analysis

### 4.5.1 Automatic Evaluation Results

Table 4.4 shows the average results for the metrics outlined in §4.3.2. From the phonetic difficulty results, we see that our formulated **PANCETTA** models score higher than the baselines. More specifically, comparing g2g (GPT-2) (0.774) vs **PANCETTA-J** (GPT-2) (0.785) and comparing g2g (GPT-J) (0.848) vs **PANCETTA-J** (GPT-J) (0.866), we see that incorporating phoneme representations indeed aids in producing more phonetically difficult sentences. This pattern also holds true for TT-Keyword, where both GPT-2 and GPT-J see increases in performance after incorporating phonemes. We also observe that **PANCETTA-P** performs reasonably well in phonetic difficulty and has the highest score of the non-GPT-J models for both TT-Prompt and TT-Keyword. In fact, for TT-Keyword, **PANCETTA-P** is able to get very close to **PANCETTA-J** (GPT-J), which is remarkable, considering that it is only using a GPT-2 model.

Meanwhile, for automatic evaluation of fluency, the style transfer models score better than the other models. This is likely because the style transfer models first generate a regular sentence, then attempt to "tongue twisterize" it. However, this is a difficult task, and there is no guarantee that the sentence can even be reasonably converted to a tongue twister. Hence, the style transfer models often fail, which results in no changes being made to the original GPT-generated sentence, thereby leading to good fluency scores when using perplexity. On the other hand, we see that

[5]https://huggingface.co/hivemind/gpt-j-6B-8bit
[6]More details about the human eval are in Appendix 4.9.

**PANCETTA-P** has the worst perplexity score. However, it is important to note that even the gold reference tongue twisters score poorly here (around the same scores as our **PANCETTA** models). This further demonstrates that tongue twisters likely are inherently less fluent in terms of utilizing typical English tokens in standard sequences, thereby resulting in worse perplexity scores. This shows that perplexity may not be the best measure of fluency for our task, and that fluency itself may not exactly correlate with the quality of a tongue twister (see §4.5.2 for more).

Lastly, for keyword relevance, most scores are close to each other. The three **PANCETTA** models have the three highest scores, indicating that **PANCETTA** is able to generate difficult tongue twisters without compromising the task at hand.

### 4.5.2 Human Evaluation Results

Table 4.5 shows the average results for the human evaluation. As with the automatic evaluations, we see an increase in phonetic difficulty when we introduce phonemes into the training process. Comparing g2g (GPT-2) and **PANCETTA-J** (GPT-2), we see an increase from 3.056 to 3.125 for TT-Prompt and from 3.847 to 3.889 for TT-Keyword. This trend also occurs for GPT-J models. Despite not being able to leverage existing GPT pre-training, **PANCETTA-P** also works very well, outperforming all non- **PANCETTA** models in all but one setting. These positive results indicate that incorporating phonetic information is indeed helpful.

In terms of phonetic difficulty, we observe that for TT-Prompt, **PANCETTA-P** works best for both GPT-2 and GPT-J, while for TT-Keyword, **PANCETTA-J** works best for both GPT-2 and GPT-J. This may be because generating from keywords is generally more difficult than completing a prompt, so **PANCETTA-J** benefits from existing pre-training. Meanwhile, in terms of fluency, g2g methods work best for 2 settings, and **PANCETTA-P** works best for 2 settings. Tongue twisters usually use words in creative and unnatural-sounding ways, and this may sometimes negatively affect fluency, so the "most fluent" sentence may not necessarily be the best tongue twister. Nevertheless, we want the **PANCETTA** outputs to still be coherent, which is indeed the case — all the fluency metrics for **PANCETTA** models are around 3.7 to 3.8. Overall, we conclude that **PANCETTA** significantly improves phonetic difficulty while maintaining a competitive level of fluency.

### 4.5.3 Qualitative Analysis

Table 4.6 shows sample generations for both TT-Prompt and TT-Keyword. We observe that both PAN-P and PAN-J are able to use a wide variety of tongue twister techniques, such as rhyme

(*grape/crepe/crate-* in ex.1 PAN-P), alliteration (*kneadle/knuckle* in ex.3 PAN-J), alternating final sounds (*land/lamb* in ex.2 PAN-P), alternating initial sounds (*six-/sheik* in ex.4 PAN-J), and repetition. They are also able to generate proper nouns to suit the sentence, such as "*Donna*" and "*Needles Nood*" in ex.3 PAN-P. They can also combine multiple such techniques in a single tongue twister, such as *stick/stock* and *land/lamb* in ex.2 PAN-P.

Comparing this with the baseline methods (g2g and Style T.), we see that the generated outputs of g2g are decent but usually very short and simple, often relying too much on alliterations. Meanwhile, the outputs for the style transfer methods are generally not tongue twisters. As discussed in §4.5.1, this is likely because it commonly fails at fully converting a regular sentence into a tongue twister.

For TT-Prompt, we obsserve that even with a non-alliterative prompt such as "If you stick" (ex.2), the **PANCETTA** models can still generate good tongue twisters, whereas the g2g method attempts to use repetition but the generated text is not that difficult to pronounce. Meanwhile, for TT-Keyword, PAN-J is able to incorporate the semantics of the words, rather than just copying the words themselves: in ex.4, PAN-J replaces "thirty/thieves" in the keywords with "sixty/sheiks". Lastly, comparing PAN-P and PAN-J, we see that PAN-J sentences generally sound smoother, while PAN-P sentences sometimes end rather abruptly ("*in a farm.*" in ex.2; "*and the Need?*" in ex.3). This is likely because PAN-P is unable to leverage existing GPT pre-training. On the other hand, this frees up PAN-P to use more diverse tongue twister techniques, such as rhymes (ex.1) and proper nouns (ex.3) which are less common in PAN-J.

## 4.6 Related Work

Automatic tongue twister generation is a largely unexplored task. Existing systems mostly use synonym replacements [189] on existing tongue twisters, which requires a large list of tongue twisters to begin with and cannot generate novel ones from scratch. Carey [17] generates tongue twisters using sound vectors, and Joshipura [78] trains an LSTM on a small tongue twister dataset, but neither are able to produce novel and semantically coherent examples. Furthermore, no methods currently exist for the TT-Keyword task.

There have been multiple studies on creative generation of various figures of speech such as similes [18], metaphors [19], and hyperbole [172]. However, these other creative linguistic constructs don't require working with another modality in the same way that tongue twister generation relies on phonemes. Among these creative lingusitic constructs, the closest ones to tongue twisters would likely be alliterations [67] and poetry [28, 54]. All of these tasks require

the application of phonetics to some extent. However, tongue twister generation goes beyond alliterations and rhymes; rather it is a mix of all these various techniques. In addition, it differs from poetry generation because poetry generation focuses on generating rhythmic verses and syllables, whereas the main focus of tongue twister generation is on phonetic difficulty.

Using phonemes in language modeling has been previously explored in the speech domain for automatic speech recognition [8, 168, 183]. In this chapter, we trained a BART model to do p2g translation. Other existing methods include expectation maximization [87], A* search [27], and Hidden Markov Models [67].

## 4.7 Conclusion

In this chapter, we studied the Constrained Creative setting of automatic tongue twister generation, and explored it under two sub-settings: TT-Prompt and TT-Keyword. To facilitate this study, we curated a dataset called *TT-Corp* of English tongue twisters through an extensive search across various sources. Inspite of our extensive data curation, the dataset we curated only had $\approx 600$ tongue twisters. Thus, we had the poor Data Availability that is a marked characteristic of this Constrained Creative class of settings being present. We first introduced the notion of a "mouth model" (see §4.1.1) that served as as the underlying Creative Story.

We then formulated **PANCETTA**, a training methodology that incorporates phoneme representations. We implemented two variants: **PANCETTA-P** (Phoneme), which trains a phoneme-based language model, and **PANCETTA-J** (Joint), which jointly incorporates both phoneme-level information and grapheme-level information during training time. Through empirical results and qualitative evaluations, we showed that incorporating phonemes as facilitated through our **Intervention** is indeed helpful in producing effective tongue twisters that are harder to pronounce while staying fluent.

While **PANCETTA** works well at generation, the generation process lacks interpretability. This is most notable when looking at the phonetic difficulty classifier. Currently, the classifier simply takes an input sentence and outputs a score; it does not identify and separately score elements of phonetic difficulty or come up with an explicit decomposition. We believe that such explicit decomposition can be very useful in the future for understanding more about tongue twisters and the "mouth model" discussed in §4.1.

## 4.8 Appendix A: Additional Details — Dataset Collection

### 4.8.1 Sources

We curate our tongue twisters from a heterogeneous mix of online sources, including but not limited to the ones listed below:

1. University of Arkansas
2. The r/tonguetwister Subreddit
3. Various AskReddit threads
4. Mondly.com
5. Uebersetzung
6. Marcus Stuart's LOL Tongue Twisters book
7. Language Avenue
8. Bilingual Monkeys
9. Pun.me
10. ESL
11. Sweetrhymes
12. EngVid
13. IvyPanda

### 4.8.2 Vetting

We then perform the following filtering steps to retain only a collection of high-quality dataset examples:

- Remove near-repetitive examples to ensure each example is unique
- Remove excessively short or meaningless examples lacking sentence structure, e.g., *blue blood, bad blood*
- Remove poems or rhymes
- Remove examples containing offensive words, racism, gender bias or other harmful and malicious language of any nature, to prevent models learnt from this data from further ingraining or amplifying such phenomena.

## 4.9 Appendix B: Evaluation Details

### 4.9.1 Human Evaluation Setup

To prevent annotator judgements for one attribute from inadvertently influencing the other, we conduct the studies for soliciting Fluency and Phonetic Difficulty scores separately.

Averaging over the 4 settings described in §4.4.2 ( TT-Keyword/TT-Prompt $\times$ GPT-2/GPT-J), a total of 20 unique AMT annotators participated in the study for Fluency, each performing 3.6 HITs on average. Annotators were compensated \$0.56 per HIT, each of which was designed to take < 2 mins on average.

Averaging over the 4 settings, 16.51 unique AMT annotators participated in the second, separate study for Phonetic Difficulty, each performing 4.36 HITs on average. Annotators were compensated \$0.56 per HIT, each of which was designed to take < 2 mins on average.

The html template including instructions, questions, and other details can be found in a file named `template.html` in our code submission zip.

### 4.9.2 Inter-Annotator Agreement (IAA) Scores

See Table 4.7 for IAA scores. To get the Spearman correlation point value for a given aspect and test instance, we compute mean pairwise Spearman correlation between the aspect values assigned to the corresponding model outputs by every pair of annotators. Specifically, we use the *scipy.stats* implementation to compute this.[7]

For Krippendorff $\alpha$, we treat each human evaluation aspect as an ordinal quantity. Specifically, we use the implementation provided by the python library *krippendorff 0.5.1*.[8]

## 4.10 Further Implementation Details

In §4.4.1, we detailed the hyperparameters used for pre-training BART, as well as for fine-tuning GPT-2, GPT-J, BART, and T5. We conduct a hyperparameter search to check which values led to the best performance. For learning rate, we tried {1e-6, 5e-6, 1e-5, 2e-5, 2e-5, 1e-4}; for batch size, we tried {2,4,8,16}; and for number of epochs, we tried {2, 5, 10, 20}. These search bounds were selected based on known commonly-used values for these models. We start with a baseline

[7] https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.spearmanr.html

[8] https://pypi.org/project/krippendorff/

model of lr=2e-5, bsz=8 and 10 epochs, and individually change each setting to investigate its effect on performance. One trial was conducted per hyperparameter setting. We use a maximum sequence length of 256. In terms of other hyperparameters, we mostly used default values which are known to work for these models. This includes the warmup steps and learning rate decays, which we detail in §4.4.1. (Note: the above hyperparameter search settings are for fine-tuning. We could not do an extensive hyperparameter search for pre-trainig due to time constraints. We ran pre-training twice to test the effect of learning rate 1e-4 vs. 5e-4, and ultimately selected 5e-4.)

Model selection was done based on the epoch with the best validation loss. We report the best validation losses for each training process below: GPT-2 pre-training=0.77, GPT-2 fine-tuning (TT-Prompt)=1.08, GPT-2 fine-tuning (TT-Keyword) =0.95, GPT-J fine-tuning (TT-Prompt)=1.23, GPT-J fine-tuning (TT-Keyword)=1.19, BART-fine-tuning (TT-Keyword) = 1.31, BART-fine-tuning (TT-Prompt) = 1.30, T5-fine-tuning (TT-Keyword) = 1.45, T5-fine-tuning (TT-Prompt) = 1.52.

All training was done on Google Colaboratory environments using V100 GPUs with 16GB memory. Pre-training GPT-2 for the p2p model took 90 minutes per epoch, and pre-training BART for the p2g model took 70 minutes per epoch. In terms of fine-tuning on *TT-Corp*, GPT-2 takes around 1 minute per epoch, GPT-J around 2 minutes per epoch, BART around 1 minute per epoch, and T5 around 1 minute per epoch.

## 4.11 Additional Qualitative Examples

In §4.5.3, we only report the best performing models for each of the methods. In Table 4.8, we extend this to all models for all the methods. We also provide a few more examples for both TT-Prompt and TT-Keyword.

| Method | Text |
|---|---|
| Inputs | (ex.1) The pirates pirated ... |
| Gold out. | The pirates pirated private property. |
| g2g | The pirates pirated the pepper pot. |
| Style T. | The pirates pirated more than $50 million in online gambling in 2013. |
| PAN-P | The pirates pirated a little grape crepe. The pirate dread cratered a little. |
| PAN-J | The pirates pirated the paired prince with a pair of paired pistols. |
| Inputs | (ex.2) If you stick ... |
| Gold out. | If you stick a stock of liquor in your locker, it is slick to stick a lock upon your stock or some joker who is slicker is going to trick you of your liquor if you fail to lock your liquor with a lock? |
| g2g | If you stick two sticks together, you get two sticks tied together. |
| Style T. | If you stick with the same line, you have different things going for you. |
| PAN-P | If you stick a stock of land in a lamb and a lamb in a farm. |
| PAN-J | If you stick a stock stick on the stock stick you stick, who's going to stick it on the stock stick you stick? |

| Method | Text |
|---|---|
| Inputs | (ex.3) needles, noodles, kneadle, don, need |
| Gold out. | I don't need your needles to kneadle noodles. |
| g2g | One noodle needs a kneeler to lean on |
| Style T. | Near the end of his reign, Leiblau had to send a delegate to Parliament, to support his friend King Gules of Northumbria. |
| PAN-P | If Donna Needles Need, who is Needles Nood, and the Need? |
| PAN-J | Don't kneadle a knuckle with a knuckle needle! |
| Inputs | (ex.4) thieves, thankful, thanked, thirty, thursday |
| Gold out. | The thirty-three thankful thieves thought that they thanked the other thirty-three thankful thieves throughout Thursday. |
| g2g | The thieve thanked the thankful thief on Thursday. |
| Style T. | Thanked by Thnx for the idea, I thought it was an idea that I wanted to do a spoof of Thankful Thankful and Thnx. |
| PAN-P | Thankful thieves thought that they thought they thrilled the throne throughout Thursday. |
| PAN-J | These sixty sheiks sent these thousand and sixty sheiks sixty sheiks thanking them for shouting these sixty sheiks sixty shouts on Thursday. |

Table 4.6: Qualitative examples for both TT-Prompt (first 2 examples) and TT-Keyword (last 2 examples). We report only the best performing model based on phonetic difficulty from automatic evaluations for each type (in brackets): literal input, gold output, g2g (GPT-J), Style Transfer (BART), **PANCETTA-P** (GPT-2+BART), and **PANCETTA-J** (GPT-J). Additional examples can be found in Appendix 4.11.

| **Metric** | **Spearman Correlation** | **Krippendorff $\alpha$** |
|---|---|---|
| Phonetic Difficulty | 0.071 | 0.092 |
| Fluency | 0.006 | 0.043 |

Table 4.7: Inter-annotator agreement scores.

| Method | Text |
|---|---|
| Inputs | (ex.1) The pirates pirated ... |
| Gold out. | The pirates pirated private property. |
| g2g (GPT2) | The pirates pirated the property. |
| g2g (GPTJ) | The pirates pirated the pepper pot. |
| Style T. (BART) | The pirates pirated more than $50 million in online gambling in 2013. |
| Style T. (T5) | The pirates pirated more than $50 million in online gambling in 2013, according to a government-backed study. |
| PAN-P | The pirates pirated a little grape crepe. The pirate dread cratered a little. |
| PAN-J (GPT2) | The pirates pirated the music and the Roman river carried the criminals off. |
| PAN-J (GPTJ) | The pirates pirated the paired prince with a pair of paired pistols. |
| Inputs | (ex.2) If you stick ... |
| Gold out. | If you stick a stock of liquor in your locker, it is slick to stick a lock upon your stock or some joker who is slicker is going to trick you of your liquor if you fail to lock your liquor with a lock? |
| g2g (GPT2) | If you stick a weight silky liquor on your wrist, you should stick a weight silky liquor on your inner wrist. |
| g2g (GPTJ) | If you stick two sticks together, you get two sticks tied together. |
| Style T. (BART) | If you stick with the same line, you have different things going for you. |
| Style T. (T5) | If you stick with the same line, you have different things going for yout. |
| PAN-P | If you stick a stock of land in a lamb and a lamb in a farm. |
| PAN-J (GPT2) | If you stick a stock of liquor in your locker it is slick to put a lock on your stock. |
| PAN-J (GPTJ) | If you stick a stock stick on the stock stick you stick, who's going to stick it on the stock stick you stick? |
| Inputs | (ex.3) I saw Susie ... |
| Gold out. | I saw Susie sitting in a shoe shine shop. |
| g2g (GPT2) | I saw Susie sitting on the sofa with a big black basket of biscuits. |
| g2g (GPTJ) | I saw Susie sitting on a buttercup. |
| Style T. (BART) | I saw Susie and then she came and took me to the kitchen where we cleaned. |
| Style T. (T5) | I saw Susie and then she came and took me to the kitchen where we cleaned up and we started talking about the new dog," she recalled. |
| PAN-P | I saw Susie sitting on a seasawe horse. and I saw haunted houses. |
| PAN-J (GPT2) | I saw Susie sitting in a speck of sheared sheep's sheep's sheep's sheep skin. |
| PAN-J (GPTJ) | I saw Susie sitting on a butter bucket with a black butter bucket sat on her butter bucket. |
| Inputs | (ex.4) needles, noodles, kneadle, don, need |
| Gold out. | I don't need your needles to kneadle noodles. |
| g2g (GPT2) | Don's dirt feet meet your earrings at the bottom of the dirt cany bore trail. |
| g2g (GPTJ) | One noodle needs a kneeler to lean on |
| Style T. (BART) | Near the end of his reign, Leiblau had to send a delegate to Parliament, to support his friend King Gules of Northumbria. |
| Style T. (T5) | Near the end of his reign, Leiblau had to send a delegate to Parliament, to support his friend King Gules of Northumbria. |
| PAN-P | If Donna Needles Need, who is Needles Nood, and the Need? |
| PAN-J (GPT2) | If you crave a soft chew toy, buy a chew toy that needs to chew. |
| PAN-J (GPTJ) | Don't kneadle a knuckle with a knuckle needle! |
| Inputs | (ex.5) thieves, thankful, thanked, thirty, thursday |
| Gold out. | The thirty-three thankful thieves thought that they thanked the other thirty-three thankful thieves throughout Thursday. |
| g2g (GPT2) | Twelve thieves today took sixty dollars from Thies' birthday party. |
| g2g (GPTJ) | The thieve thanked the thankful thief on Thursday. |
| Style T. (BART) | Thanked by Thnx for the idea, I thought it was an idea that I wanted to do a spoof of Thankful Thankful and Thnx. |
| Style T. (T5) | Thanked by Thnx for the idea, I thought it was an idea that I wanted to do a spoof of Thankful Thankful and Thnx, so it went to the video website, Twitter, and did some research on a T shirt. |
| PAN-P | Thankful thieves thought that they thought they thrilled the throne throughout Thursday. |
| PAN-J (GPT2) | I'm grateful tonight for thanking the valiant brave thieves. |
| PAN-J (GPTJ) | These sixty sheiks sent these thousand and sixty sheiks sixty sheiks thanking them for shouting these sixty sheiks sixty shouts on Thursday. |

Table 4.8: Additional qualitative examples for both TT-Prompt (first 3) & TT-Keyword (last 3): literal input, gold output, g2g (GPT-2), g2g (GPT-J), Style Transfer (BART), Style Transfer (T5), **PANCETTA-P** (GPT-2+BART), **PANCETTA-J** (GPT-2), and **PANCETTA-J** (GPT-J).

# Chapter 5

# Stylistic Surface Transduction To Shakespearize Modern English (EMNLP 2017'WS)

> Most of us in speaking and writing English use only one pronoun of address; we say 'you' to many persons and 'you' to one person. The pronoun 'thou' is reserved, nowadays, to prayer and naive poetry, but in the past it was the form of familiar address to a single person. At that time 'you' was the singular of reverence and of polite distance, and also the invariable plural.
>
> R.Brown & A. Gilman, '*The Pronouns of Power & Solidarity*', 1960

As users get ever more habituated to using, interacting and even co-authoring with NLG systems, there is increasing expectation on them to exhibit *consistent personality* [165] and also be *accomodative* [161] towards *user preferences* and *situation of use*. Together, one can think of these as aspects of *target style*. Hence, NLG systems should be able to transfer their content to match aspect values for each aspect of target style. From the perspective of Halliday's SFL, style transfer can be seen as having the changing of *extra-textual aspects* as its communicative goal, while keeping the textual aspect constant. These aspects could be either interpersonal or

ideational in nature. *Diachronic register* of language is one example of an interpersonal aspect, being determined by the author of a text as well as the historical period in which the author resides. Specifically, the term diachronic refers to the perspective of language as a changing variable that evolves through *chronos* i.e., time. The diachronic register of a language simply denotes its state at a given period in history. For instance, consider the two variants *I stand on sudden haste* and *I am in a rush*. Though a native English speaker would likely understand both, and understand both of these to means the same thing in ideational terms, they would certainly find the first one a strange way of conveying the same idea, and rightly so, for it is how Shakespeare originally said it, with the latter one being a Modern English paraphrase from Sparknotes.com.

In this chapter, we study the **Constrained Creative** NLG setting of transferring the style of a given sentence authored in contemporary English e.g., *I am in a rush*, to the style of William Shakespeare, who wrote in the Early Modern English prevalent in Elizabethan times, such as e.g., *I stand on sudden haste*.

The **CG Definition/Constraints** for this setting are: Given an English source sentence, transduce it lexico-syntactically to sound like Early Modern English while preserving the source's original meaning.

In terms of **Data Availability**, we have access to $\approx$ 15,000 parallel source-target pairs, as detailed in §5.2. This is an order of magnitude lower than the typical counts of parallel data used to train strong machine translation models using attentional sequence-to-sequence transducers [5]. However, additionally, we also have access to a noisy, sparse dictionary $L_{pairwise}$ of $\approx$ 1000 source $\rightarrow$ target lexical correspondences, e.g., *thou* $\rightarrow$ you, *fetches* $\rightarrow$ excuses etc.

We first posit a **Creative Story** in §5.1.1, detailing how a human speaker would use underlying surface realization correspondences and shared linguistic structure when doing this task. This creative story in turn motivates the Interventions below:
i) We explicitly provide source $\rightarrow$ target surface realization correspondences to our model through incorporating $L_{pairwise}$ into word representations. The "word representation" here is the shared, jointly pretrained space in which both source and target words are embedded; we also empirically show that having a shared embedding space is better than having separate ones, as also that joint pretraining is beneficial.
ii) We also intervene into the model architecture, equipping its decoder with the ability to copy over words from the source sentence besides generating them from the vocabulary.

Specifically, we explore automated methods to transfer text from modern English to Shakespearean English using a neural model with pointers to enable copy action. To ameliorate the deficient learning of embeddings due to a limited amount of parallel data, we pretrain embeddings

of words by leveraging $L_{pairwise}$ mapping Shakespearean words to modern English words as well as additional text. Our methods achieve a BLEU score of $31+$, an improvement of $\approx 6$ points over the strongest baseline. We publicly release our code to foster further research in this area. [1]

## 5.1 Introduction

Given a source text, human speakers/authors/content moderators often morph it using a variety of lexical and grammatical transformations, adjusting the degree of formality, usage of catchy phrases, and other such stylistic changes. Their non-textual subgoals are, for example, to make it more appealing. For instance, assuming a "make more appealing" subgoal, different text styles appeal to different target user segments [153] [86] [154]. Millenials may find text using social media slang to be more appealing, while middle-aged New Yorkers may find the inclusion of Yiddish and Italian slang to be so. Thus there is a need to effectively adapt text to different target styles. However, manually transforming text to a desired style can be a tedious process.

There have been increased efforts towards machine assisted text content creation and editing through automated methods for summarization [151], brand naming [64], text expansion [164], etc. However, there is a dearth of automated solutions for adapting text quickly to different styles. We consider the problem of transforming text written in modern English text to Shakepearean style English. For the sake of brevity and clarity of exposition, we henceforth refer to the *Shakespearean* sentences/side as *Original* and the modern English paraphrases as *Modern*.

Unlike more traditional domain or style transfer settings e.g., formality, our task is distinguished by the fact that the two styles employ diachronically disparate registers of English — one style uses the contemporary language while the other uses *Early Modern English* [2] from the *Elizabethan Era*(1558-1603). Although *Early Modern English* is not classified as a different language (unlike *Old English* and *Middle English*), it does have novel words (*acknown* and *belike*), novel grammatical constructions (two *second person* forms: *thou* (informal) and *you* (formal) [16]), semantically drifted senses (e.g., *fetches* is a synonym of *excuses*) and non-standard orthography [140]. Additionally, there is a domain difference since the Shakespearean play sentences are from a dramatic screenplay whereas the *parallel* modern English sentences are meant to be simplified explanation for high-school students.

[1] https://github.com/harsh19/Shakespearizing-Modern-English
[2] https://en.wikipedia.org/wiki/Early_Modern_English

| No | Type | Text |
|---|---|---|
| 1 | MODERN | Oh my, my bones ache so much |
| | ORIGINAL | Fie, how my bones ache ! |
| | COPY | fie, how my bones ache ! |
| | SIMPLES2S | you'll be, sir, what the bones are tired . |
| | STAT | Oh my, my bones ache so much . |
| 2 | MODERN | I am in a rush . |
| | ORIGINAL | I stand on sudden haste . |
| | COPY | i stand on sudden haste . |
| | SIMPLES2S | i'm stand right here . |
| | STAT | I am in a Fly |
| 3 | MODERN | Give my compliments to your lady |
| | ORIGINAL | Commend me to thy lady |
| | COPY | commend me to your lady |
| | SIMPLES2S | give my regards to your lady |
| | STAT | give my praises to your lady |
| 4 | MODERN | Showing mercy by pardoning killers only causes more murders . |
| | ORIGINAL | Mercy but murders, pardoning those that kill . |
| | COPY | mercy but murders, those those who kill us . |
| | SIMPLES2S | but except the murders to those murders to kill you . |
| | STAT | of mercy by pardoning killers causes more dire. |
| 5 | MODERN | Holy Saint Francis, this is a drastic change ! |
| | ORIGINAL | Holy Saint Francis, what a change is here ! |
| | COPY | holy saint francis, what a change is here ! |
| | SIMPLES2S | it's the holy flute, what's the changed ! |
| | STAT | Holy Saint Francis, this is a drastic change ! |
| 6 | MODERN | was that my father who left here in such a hurry ? |
| | ORIGINAL | Was that my father that went hence so fast ? |
| | COPY | was that my father that went went so fast ? |
| | SIMPLES2S | was that my father was so that ? |
| | STAT | was that my father that left here in such a haste ? |
| 7 | MODERN | Give me one kiss and I'll go down . |
| | ORIGINAL | One kiss, and I'll descend . |
| | COPY | one kiss me, and I'll descend . |
| | SIMPLES2S | one kiss,and I come down . |
| | STAT | Give me a kiss, and I'll go down . |
| 8 | MODERN | then the window lets day in, and life goes out the window . |
| | ORIGINAL | Then, window, let day in and life out . |
| | COPY | then, window out, and day life . |
| | SIMPLES2S | then she is just a life of life, let me life out of life . |
| | STAT | then the window will let day in, and life out . |

Table 5.1: Examples from dataset showing modern paraphrases (MODERN) from the learning resource Sparknotes.com of few sentences from Shakespeare's plays (ORIGINAL). We also show transformation of modern text to Shakespearean text from our models (COPY, SIMPLES2S and STAT).

### 5.1.1 Creative Story

Consider how an idealized human speaker, for example, a native English speaking high school student with a basic familarity of reading literary fiction would go about accomplishing the task. Perhaps, the speaker would already have a working familiarity with Early Modern English having read Shakespeare in the past. If not, they would first familiarize themselves with the dialect by skimming through a few pairs of Shakespeare's sentences and their modern English explanations. They would also skim through some entries of $L_{pairwise}$ to disambiguate any frequent, confusing Early Modern English words they would have run into, as also to quickly familiarize themselves with the range of Early Modern English words which have either gone our of use or diverged away in form and meaning. Having acquired a basic internal cognitive model of Early Modern English and how it relates back to today's English, the speaker would now get down to their assigned task of style transferring new sentence examples to Early Modern English. A rough analogy within a machine learning setup would be a pretraining step that learns a shared space where word types from the set union of Early Modern and Modern English vocabularies are jointly embedded.

Given a new sentence in contemporary English, the speaker would read through it and identify for focussing particular words and phrases which are: a) Critical to preserving the meaning and likely to remain unchanged e.g., key action verbs like *marched*, named entities such as *Wormwood Forest* etc, and b) Likely to require changes while transferring to Early Modern English e.g. the word *excuses* or new age idioms such as *like a tracer bullet*. Having carefully read the sentence, they would now actually rewrite it into Early Modern English, directly copying over certain segments (such as those identified in point a) earlier), while replacing and rewriting some others (such as those identified in point b)). A rough analogy to this step within a machine learning setup would be a sequence-to-sequence transducer equipped with a copy/pointer component [113], whose decoder can choose between: a) Directly copying over words from the input source sentence, and b) Generating words afresh from the entire vocabulary

### 5.1.2 Prior Approaches

Some prior work in this domain leverages a language model for the target style, achieving transformation either using phrase tables [185] or by inserting relevant adjectives and adverbs [153]. Such works have limited scope in the type of transformations that can be achieved. Firstly, it is difficult for human curated canned phrase table resources to cover the combinatorially increasing number of phrase pairs. Secondly, phrase tables cannot take context into account

when doing phrase → phrase replacements. Moreover, statistical and rule MT based systems do not provide a direct mechanism to a) share word representation information between source and target sides b) incorporating constraints between words into word representations in end-to-end fashion. Neural sequence-to-sequence models, on the other hand, provide such flexibility. They provide direct mechanisms to handle all of these — Sharing source and target embeddings to share word-representation information, pretraining to leverage external information, and adding constraints to word representations using [39].

### 5.1.3 Contributions

In this chapter, our main contributions are as follows:

- We use a sentence level sequence to sequence neural model with a pointer network component to enable direct copying of words from input. We demonstrate that this method performs much better than prior phrase translation based approaches for transforming *Modern* English text to *Shakespearean* English.
- We leverage a dictionary providing mapping between Shakespearean words and modern English words to retrofit pre-trained word embeddings. Incorporating this extra information enables our model to perform well in spite of small size of parallel data.

The rest of the chapter is organized as follows. We first provide a brief analysis of our dataset in (§5.2). We then elaborate on details of our methods in (§5.3, §5.4, §5.5, §5.6). We then discuss experimental setup and baselines in (§5.7). Thereafter, we discuss the results and observations in (§5.8). We conclude with discussions on related work (§5.9) and future directions (§5.10).

## 5.2 Dataset

Our dataset is a collection of line-by-line modern paraphrases for 16 of Shakespeare's 36 plays (*Antony & Cleopatra*, *As You Like It*, *Comedy of Errors*, *Hamlet*, *Henry V* etc.) from the educational site *Sparknotes*[3]. This dataset was compiled by Xu et al. [184, 185] and is freely available on github.[4] 14 plays covering 18,395 sentences form the training data split. We kept 1218 sentences from the play *Twelfth Night* as validation data set. The last play, *Romeo and Juliet*, comprising 1462 sentences, forms the test set.

[3] www.sparknotes.com
[4] http://tinyurl.com/ycdd3v6h

| | *Original* | *Modern* |
|---|---|---|
| # Word Tokens | 217K | 200K |
| # Word Types | 12.39K | 10.05K |
| Average Sentence Length | 11.81 | 10.91 |
| Entropy (Type.Dist) | 6.15 | 6.06 |
| ∩ Word Types | 6.33K | |

Table 5.2: Dataset Statistics

### 5.2.1 Examples

Table 5.1 shows some parallel pairs from the test split of our data, along with the corresponding target outputs from some of our models. *Copy* and *SimpleS2S* refer to our best performing attentional S2S models with and without a *Copy* component respectively. *Stat* refers to the best statistical machine translation baseline using off-the-shelf GIZA++ aligner and MOSES. We can see through many of the examples how direct copying from the source side helps the *Copy* generates better outputs than the *SimpleS2S*. The approaches are described in greater detail in (§5.3) and (§5.7).

### 5.2.2 Analysis

Table 5.2 shows some statistics from the training split of the dataset. In general, the *Original* side has longer sentences and a larger vocabulary. The slightly higher entropy of the *Original* side's frequency distribution indicates that the frequencies are more spread out over words. Intuitively, the large number of shared word types indicates that sharing the representation between *Original* and *Modern* sides could provide some benefit.

## 5.3 Method Overview

The Overall architecture of our system is shown in Figure 5.1. We use a bidirectional LSTM to encode the input modern English sentence. Our decoder side model is a mixture model of RNN module amd pointer network module. The two individual modules share the attentions weights over encoder states, although it is not necessary to do so. The decoder RNN predicts probability distribution of next word over the vocabulary, while the pointer model predicts the probability distribution over words in input. The two probabilities undergo a weighted addition, the weights themselves computed based on previous decoder hidden state and the encoder outputs.

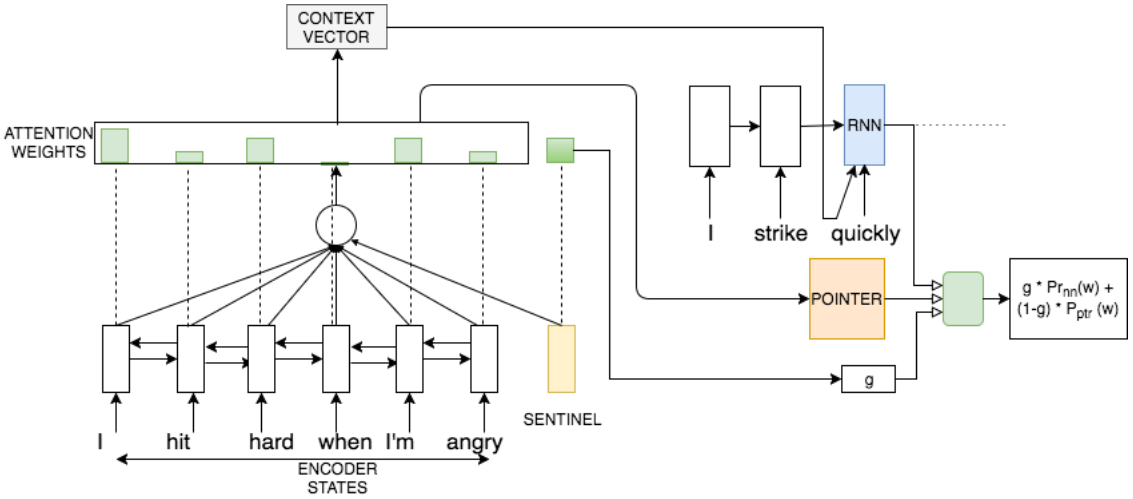



Figure 5.1: Depiction of our overall architecture (showing decoder step 3). Attention weights are computed using previous decoder hidden state $h_2$, encoder representations, and sentinel vector. Attention weights are shared by decoder RNN and pointer models. The final probability distribution over vocabulary comes from both the decoder RNN and the pointer network. Similar formulation is used over all decoder steps

Let $\mathbf{x}, \mathbf{y}$ be the some input - output sentence pair in the dataset. Both input $\mathbf{x}$ as well as output $\mathbf{y}$ are sequence of tokens. $\mathbf{x} = \mathbf{x}_1\mathbf{x}_2...\mathbf{x}_{T_{enc}}$, where $T_{enc}$ represents the length of the input sequence $\mathbf{x}$. Similarly, $\mathbf{y} = \mathbf{y}_1\mathbf{y}_2...\mathbf{y}_{T_{dec}}$. Each of $\mathbf{x}_i$, $\mathbf{y}_j$ is a token from the vocabulary.

## 5.4 Token embeddings

Each token in vocabulary is represented by a $M$ dimensional embedding vector. Let vocabulary $V$ be the union of modern English and Shakepearean vocabularies i.e. $V = V_{shakespeare} \cup V_{modern}$. $E_{enc}$ and $E_{dec}$ represent the embedding matrices used by encoder and decoder respectively ( $E_{enc}, E_{dec} \in \mathbb{R}^{|V| \times M}$ ). We consider union of the vocabularies for both input and output embeddings because many of the tokens are common in two vocabularies, and in the best performing setting we share embeddings between encoder and decoder models. Let $E_{enc}(t)$, represent encoder side embeddings of some token $t$. For some input sequence $\mathbf{x}$, $E_{enc}(\mathbf{x})$ is given as $(E_{enc}(\mathbf{x}_1), E_{enc}(\mathbf{x}_2), ...)$.

### 5.4.1 Pretraining of embeddings

Learning token embeddings from scratch in an end-to-end fashion along with the model greatly increases the number of parameters. To mitigate this, we consider pretraining of the token

embeddings. We pretrain our embeddings on all training sentences. We also experiment with adding additional data from the Penn Tree Bank (PTB) [110] for better learning of embeddings. Additionally we leverage a dictionary mapping tokens from Shakespearean English to modern English.

We consider four distinct strategies to train the embeddings. In the cases where we use external text data, we first train the embeddings using both the external data and training data, and then for the same number of iterations on training data alone, to ensure adaptation. Note that we do not directly use off-the-shelf pretrained embeddings such as *GloVe* [130] and *Word2Vec* [115] since we need to learn embeddings for novel word forms (and also different word senses for extant word forms) on the *Original* side.

#### Plain

This method is the simplest pre-training method. Here, we do not use any additional data, and train word embeddings are trained on the union of *Modern* and *Original* sentences.

#### PlainExt

In this method, we add all the sentences from the external text source (*PTB*) in addition to sentences in training split of our data.

#### Retro

We leverage a dictionary $L$ of approximate *Original* $\rightarrow$ *Modern* word pairs [184, 185], crawled from `shakespeare-words.com`, a source distinct from Sparknotes. We explicitly add the two *2nd persons* and their corresponding forms (thy, thou, thyself ,etc.) which are very frequent but not present in $L$. The final dictionary we use has 1524 pairs. Faruqui et al. [39] proposed a *retrofitting* method to update a set of word embeddings to incorporate pairwise similarity constraints. Given a set of embeddings $p_i \in P$, a vocabulary $V$, and a set $C$ of pairwise constraints $(i, j)$ between words, retrofitting tries to learn a new set of embeddings $q_i \in Q$ to minimize the following objective:

$$f(Q) = \delta \sum_{i=1}^{i=|V|} (p_i - q_i)^2 + \omega \sum_{(i,j)\in C} (q_i - q_j)^2 \tag{5.1}$$

We use their off-the-shelf implementation [5] to encode the dictionary constraints into our pretrained embeddings, setting $C = L$ and using suggested default hyperparameters for $\delta$, $\omega$ and number of iterations.

**RetroExt**

This method is similar to *Retro*, except that we use sentences from the external data (*PTB*) in addition to training sentences.

We use **None** to represent the settings where we do not pretrain the embeddings.

### 5.4.2 Fixed embeddings

Fine-tuning pre-trained embeddings for a given task may lead to *overfitting*, especially in scenarios with small amount of supervised data for the task [108]. This is because embeddings for only a fraction of vocabulary items get updated, leaving the embeddings unchanged for many vocabulary items. To avoid this, we consider fixed embeddings pretrained as per the procedures described earlier. While reporting results in Section (§5.8), we separately report results for fixed (*FIXED*) and trainable (*VAR*) embeddings, and observe that keeping embeddings fixed leads to better performance.

## 5.5 Method Description

In this section we give details of the various modules in the devised neural model.

### 5.5.1 Encoder model

Let $\overrightarrow{LSTM_{enc}}$ and $\overleftarrow{LSTM_{enc}}$ represent the forward and reverse encoder. $\mathbf{h}_t^{\overrightarrow{enc}}$ represent hidden state of encoder model at step $t$ ($\mathbf{h}_t^{\overrightarrow{enc}} \in \mathbb{R}^H$). The following equations describe the model:

$$\mathbf{h}_0^{\overrightarrow{enc}} = \overrightarrow{0}, \mathbf{h}_{|x|}^{\overleftarrow{enc}} = \overrightarrow{0} \tag{5.2}$$

$$\mathbf{h}_t^{\overrightarrow{enc}} = \overrightarrow{LSTM_{enc}}(\mathbf{h}_{t-1}^{enc}, E_{enc}(\mathbf{x}_t)) \tag{5.3}$$

$$\mathbf{h}_t^{\overleftarrow{enc}} = \overleftarrow{LSTM_{enc}}(\mathbf{h}_{t+1}^{enc}, E_{enc}(x_t)) \tag{5.4}$$

$$\mathbf{h}_t^{enc} = \mathbf{h}_t^{\overrightarrow{enc}} + \mathbf{h}_t^{\overleftarrow{enc}} \tag{5.5}$$

[5] github.com/mfaruqui/retrofitting

### 5.5.2 Attention

Let $\mathbf{h}_t^{dec}$ represent the hidden state of the decoder LSTM at step $t$. Let $E_{dec}(\mathbf{y}_{t-1})$ represent the decoder side embeddings of previous step output. We use special $START$ symbol at $t = 1$.

We first compute a query vector, that is a linear transformation of $\mathbf{h}_{t-1}^{dec}$. A sentinel vector $\mathbf{s} \in \mathbb{R}^H$ is concatenated with the encoder states to create $F_{att} \in \mathbb{R}^{(T_{enc}+1)\times H}$, where $T_{enc}$ represents the number of tokens in encoder input sequence $\mathbf{x}$. A normalized attention weight vector $\boldsymbol{\alpha}^{norm}$ is computed. The value $g$, which corresponds to attention weight over the sentinel vector, represents the weight given to the decoder RNN module while computing output probabilties.

$$\mathbf{q} = \mathbf{h}_{t-1}^{dec} W_q \qquad W_q \in \mathbb{R}^{H\times H} \tag{5.6}$$

$$F_{att} = concat(\mathbf{h}_{1..T_{enc}}^{enc}, \mathbf{s}) \qquad F_{att} \in \mathbb{R}^{(T_{enc}+1)\times H} \tag{5.7}$$

$$\boldsymbol{\alpha}_i = \sum_{j=1}^{H} (tanh(F_{att}^{(ij)}\, \mathbf{q}_j)) + \mathbf{b}_i \qquad \boldsymbol{\alpha}_i, \mathbf{b}_i \in \mathbb{R} \tag{5.8}$$

$$\boldsymbol{\alpha}^{norm} = softmax(\boldsymbol{\alpha}) \qquad \boldsymbol{\alpha}^{norm} \in \mathbb{R}^{T_{enc}+1} \tag{5.9}$$

$$\boldsymbol{\beta} = \boldsymbol{\alpha}_{1,2,\ldots,T_{enc}}^{norm} \qquad \boldsymbol{\beta} \in \mathbb{R}^{T_{enc}} \tag{5.10}$$

$$g = \boldsymbol{\alpha}_{T_{enc}+1}^{norm} \qquad g \in \mathbb{R} \tag{5.11}$$

### 5.5.3 Pointer model

As noted earlier, a pair of corresponding *Original* and *Modern* sentences have significant vocabulary overlap. Moreover, there are lot of proper nouns and rare words that might not be predicted by a sequence to sequence model. To rectify this, pointer networks have been used to enable copying of tokens from input directly [113]. The pointer module provides location based attention, and output probability distribution due to pointer network module can be expressed as follows:

$$P_t^{PTR}(w) = \sum_{\mathbf{x}_j = w} (\boldsymbol{\beta}_j) \tag{5.12}$$

### 5.5.4 Decoder RNN

Summation of encoder states weighed by corresponding attention weights yields the context vector. Output probabilities over vocabulary as per the decoder LSTM module are computed as follows:

$$\mathbf{c}_t = \sum_{i=1}^{T_{enc}} \boldsymbol{\beta}_i \, \mathbf{h}_i^{enc} \tag{5.13}$$

$$\mathbf{h}_t^{dec} = LSTM(\mathbf{h}_{t-1}^{dec}, [\text{concat}(E_{dec}(\mathbf{y}_{t-1}), \mathbf{c}_t)]) \tag{5.14}$$

$$P_t^{LSTM} = \text{softmax}(W_{out}[\text{concat}(\mathbf{h}_t^{dec}, \mathbf{c}_t)] + \mathbf{b}^{out}) \tag{5.15}$$

During training, we feed the ground truth for $\mathbf{y}_{t-1}$, whereas while making predictions on test data, predicted output from previous step is used instead.

### 5.5.5 Output prediction

Output probability of a token $w$ at step $t$ is a weighted sum of probabilities from decoder LSTM model and pointer model given as follows:

$$P_t(w) = g \times P_t^{LSTM}(w) + (1 - g) \times P_t^{PTR}(w) \tag{5.16}$$

$P_t^{PTR}(w)$ takes a non-zero value only if $w$ occurs in the input sequence, otherwise it is $0$. Forcing $g = 0$ would correspond to not having a *Copy* component, reducing the model to a plain attentional S2S model, that we refer to as *SimpleS2S*.

## 5.6 Loss functions

Cross entropy loss is used to train the model. For a data point $(\mathbf{x}, \mathbf{y}) \in \mathcal{D}$ and predicted probability distributions $P_t\left(w\right)$ over the different words $w \in \mathbf{V}$ for each time step $t \in \{1, \ldots, T_{dec}\}$, the loss is given by

$$-\sum_{t=1}^{T_{dec}} \log p\left(P_t\left(\mathbf{y}_t\right)\right) \tag{5.17}$$

**Sentinel Loss (SL):** Following from work by [113], we consider additional sentinel loss. This loss function can be considered as a form of *supervised attention*. Sentinel loss is given as follows:

$$-\sum_{t=1}^{T_{dec}} \log(g^{(t)} + \sum_{x_j=y_t} (\beta_j^{(t)})) \tag{5.18}$$

Note that the above formulation of sentinel loss is also taken from [113]. We report the results demonstrating the impact of including the sentinel loss function (+SL).

## 5.7 Experiments

In this section we describe the experimental setup and evaluation criteria used.

### 5.7.1 Preprocessing

We lowercase sentences and then use NLTK's PUNKT tokenizer [85] to tokenize all sentences into their constituent words (tokens). The *Original* side has certain characters like æ that are not extant in today's English language. We map these characters to the closest equivalent character(s) used today (e.g., æ→ ae)

### 5.7.2 Baseline Methods

#### As-it-is

Since both source and target side are English, just replicating the input on the target side is a valid and competitive baseline, producing a BLEU of 21+.

#### Dictionary

Xu et al. [185] provide a dictionary mapping between large number of Shakespearean and modern English words. We augment this dictionary with pairs corresponding to the 2nd person thou (*thou*, *thy*, *thyself*) since these common tokens were not present.

Directly using this dictionary to perform word-by-word replacement is another admittable baseline. As was noted by Xu et al. [185], this baseline actually performs worse than *As-it-is*. This could be due to its performing aggressive replacement without regard for word context. Moreover, a dictionary cannot easily capture one-to-many mappings as well as long-range dependencies.[6]

[6]thou-thyself and you-yourself

**Off-the-shelf SMT**

To train statistical machine translation (*SMT*) baselines, we use publicly available open-source toolkit MOSES [90], along with the GIZA++ word aligner [122], as was done in [185]. For training the target-side LM component, we use the *lmplz* toolkit within MOSES to train a 4-gram LM. We also use *MERT* [122], available as part of MOSES, to tune on the validation set.

For fairness of comparison, it is necessary to use the pairwise dictionary and *PTB* while training the SMT models as well. The most obvious way for this is to use the dictionary and *PTB* as additional training data for the alignment component and the target-side LM respectively. We experiment with several SMT models, ablating for the use of both *PTB* and dictionary. In 5.8, we only report the performance of the best of these approaches.

### 5.7.3 Evaluation

Our primary evaluation metric is *BLEU* [125]. We compute *BLEU* using the freely available and very widely used perl script[7] from the MOSES decoder.

We also report *PINC* [20], a metric which originates from paraphrase evaluation literature and evaluates how much the target side paraphrases resemble the source side. Given a source sentence $s$ and a target side paraphrase $c$ generated by the system, *PINC(s,c)* is defined as

$$PINC(s,c) = 1 - \frac{1}{N}\sum_{n=1}^{n=N}\frac{|Ngram(c,n) \cap Ngram(s,n)|}{|Ngram(c,n)|}$$

where $Ngram(x,n)$ denotes the set of all n-grams of length $n$ in sentence $x$, and $N$ is the maximum length of ngram considered. We set $N = 4$. Higher the *PINC*, greater the novelty of paraphrases generated by the system. Note, however, that PINC *does not measure fluency* of generated paraphrases. Moreover, it cannot be used to compare against references. Hence, it rewards all changes similarly irrespective of their fluency as well as adequacy, merely proportional to the extent of ngrams edited. As a result, it can merely be used as an auxiliary metric.

**Limitations of Current Metrics**

We acknowledge that BLEU and PINC are both deficient in various aspects, e.g., not considering higher order ngrams, being insensitive to paraphrasing and near-synonymy, being incapable of assigning partial credit for non-exactly matching string forms etc. Evaluation metrics that

[7] http://tinyurl.com/yben45gm

compare the embedding representations of generated and gold standard sentences would have been more appropriate for this purpose. More definitively, a thorough evaluation of the generated sentences on fluency, semantic correctness and other typical aspects of text quality by skilled human annotators capable of reading and understanding Shakespearean English would have been ideal.

### 5.7.4 Training and Parameters

We use a minibatch-size of 32 and the *ADAM* optimizer [84] with learning rate $0.001$, momentum parameters $0.9$ and $0.999$, and $\epsilon = 10^{-8}$. All our implementations are written in Python using Tensorflow 1.1.0 framework.

For every model, we experimented with two configurations of embedding and LSTM size: $S$ (128-128), $ME$ (192-192) and $L$ (256-256). Across models, we find that the $ME$ configuration performs better in terms of highest validation BLEU. We also find that larger configurations (384-384 and 512-512) fail to converge or perform very poorly.[8] Here, we report results only for the $ME$ configuration for all the models. For all our models, we picked the best saved model over 15 epochs that has the highest validation BLEU.

### 5.7.5 Decoding

At test-time we use greedy decoding to find the most likely target sentence.[9] We also experiment with a post-processing strategy that replaces *UNKs* in the target output with the highest aligned (maximum attention) source word. We find that this gives a small jump in *BLEU* of about 0.1-0.2 for all neural models.[10] Our best model, for instance, gets a jump of 0.14 to reach a BLEU of ***31.26*** from 31.12.

## 5.8 Results

The results in Table 5.3 confirm most of our hypotheses about the right architecture for this task.

- **Copy component**: We can observe from Table 5.3 that the various *Copy* models each outperform their *SimpleS2S* counterparts by at least 7-8 BLEU points.

[8] This is expected given the small parallel data
[9] Empirically, we observed that beam search does not give improvements for our task
[10] Since effect is small and uniform, we report BLEU before post-processing in Table 5.3

- **Retrofitting dictionary constraints**: The *Retro* configurations generally outperform their corresponding *Plain* configurations. For instance, our best configuration *Copy.Yes.RetroExtFixed* gets a better BLEU than *Copy.Yes.PlainExtFixed* by a margin of at least 11.
- **Sharing Embeddings**: Sharing source and target side embeddings benefits all the *Retro* configurations, although it slightly deteriorates performance (about 1 BLEU point) for some of the *Plain* configurations.
- **Fixing Embeddings**: *Fixed* configurations always perform better than corresponding *Var* ones (save some exceptions). For instance, *Copy.Yes.RetroExtFixed* get a BLEU of 31.12 compared to 20.95 for *Copy.Yes.RetroExtVar*. Due to fixing embeddings, the former has just half as many parameters as the latter (5.25M vs 9.40M)
- **Effect of External Data**: Pretraining with external data *Ext* works well along with retrofitting *Retro*. For instance, *Copy.Yes.RetroExtFixed* gets a BLEU improvement of 2+ points over *Copy.Yes.RetroFixed*
- **Effect of Pretraining**: For the *SimpleS2S* models, pre-training adversely affects BLEU. However, for the *Copy* models, pre-training leads to improvement in BLEU. The simplest pretrained *Copy* model, *Copy.No.PlainVar* has a BLEU score 1.8 higher than *Copy.No.NoneVar*.
- **PINC scores**: All the neural models have higher PINC scores than the statistical and dictionary approaches, which indicate that the target sentences produced differ more from the source sentences than those produced by these approaches.
- **Sentinel Loss:** Adding the sentinel loss does not have any significant effect, and ends up reducing BLEU by a point or two, as seen with the *Copy+SL* configurations.

### 5.8.1 Qualitative Analysis

Table 5.1 presents model outputs for some test examples. In general, the *Copy* model outputs resemble the ground truth more closely compared to *SimpleS2S* and *Stat*. In some cases, it faces issues with repetition (Examples 4 and 6) and fluency (Example 8).

Figure 5.2 shows the attention matrices from our best *Copy* model (*Copy.Yes.RetroExtFixed*) and our best *SimpleS2S* model (*SimpleS2S.Yes.Retrofixed*) respectively for the same input test sentence. Without an explicit *Copy* component, the *SimpleS2S* model cannot predict the words *saint* and *francis*, and drifts off after predicting incorrect word *flute*.

| Model | Sh | Init | BLEU | PINC |
|---|---|---|---|---|
| AS-IT-IS | - | - | 21.13 | 0.0 |
| DICTIONARY | - | - | 17.00 | 26.64 |
| STAT | - | - | **24.39** | 32.30 |
| SIMPLES2S | × | $NoneVar$ | 11.66 | 85.61 |
| | × | $PlainVar$ | 9.27 | 86.52 |
| | × | $PlainExtVar$ | 8.73 | 87.17 |
| | × | $RetroVar$ | 10.57 | 85.06 |
| | × | $RetroExtVar$ | 10.26 | 83.83 |
| | ✓ | $NoneVar$ | 11.17 | 84.91 |
| | ✓ | $PlainVar$ | 8.78 | 85.57 |
| | ✓ | $PlainFixed$ | 8.73 | 89.19 |
| | ✓ | $PlainExtVar$ | 8.59 | 86.04 |
| | ✓ | $PlainExtFixed$ | 8.59 | 89.16 |
| | ✓ | $RetroVar$ | 10.86 | 85.58 |
| | ✓ | $RetroFixed$ | 11.36 | 85.07 |
| | ✓ | $RetroExtVar$ | 11.25 | 83.56 |
| | ✓ | $RetroExtFixed$ | **10.86** | 88.80 |
| COPY | × | $NoneVar$ | 18.44 | 83.68 |
| | × | $PlainVar$ | 20.26 | 81.54 |
| | × | $PlainExtVar$ | 20.20 | 83.38 |
| | × | $RetroVar$ | 21.25 | 81.18 |
| | × | $RetroExtVar$ | 21.57 | 82.89 |
| | ✓ | $NoneVar$ | 22.70 | 81.51 |
| | ✓ | $PlainVar$ | 19.27 | 83.87 |
| | ✓ | $PlainFixed$ | 21.20 | 81.61 |
| | ✓ | $PlainExtVar$ | 20.76 | 83.17 |
| | ✓ | $PlainExtFixed$ | 19.32 | 82.38 |
| | ✓ | $RetroVar$ | 22.71 | 81.12 |
| | ✓ | $RetroFixed$ | **28.86** | 80.53 |
| | ✓ | $RetroExtVar$ | 20.95 | 81.94 |
| | ✓ | $RetroExtFixed$ | **31.12** | 79.63 |
| COPY+SL | × | $NoneVar$ | 17.88 | 83.70 |
| | × | $PlainVar$ | 20.22 | 81.52 |
| | × | $PlainExtVar$ | 20.14 | 83.46 |
| | × | $RetroVar$ | 21.30 | 81.22 |
| | × | $RetroExtVar$ | 21.52 | 82.86 |
| | ✓ | $NoneVar$ | 22.72 | 81.41 |
| | ✓ | $PlainVar$ | 21.46 | 81.39 |
| | ✓ | $PlainFixed$ | 23.76 | 81.68 |
| | ✓ | $PlainExtVar$ | 20.68 | 83.18 |
| | ✓ | $PlainExtFixed$ | 22.23 | 81.71 |
| | ✓ | $RetroVar$ | 22.62 | 81.15 |
| | ✓ | $RetroFixed$ | 27.66 | 81.35 |
| | ✓ | $RetroExtVar$ | 24.11 | 79.92 |
| | ✓ | $RetroExtFixed$ | 27.81 | 84.67 |

Table 5.3: Test BLEU results. *Sh* denotes encoder-decoder embedding sharing (*No*=×,*Yes*=✓) . *Init* denotes the manner of initializing embedding vectors. The *-Fixed* or *-Var* suffix indicates whether embeddings are fixed or trainable. COPY and SIMPLES2S denote presence/absence of *Copy* component. +SL denotes sentinel loss.

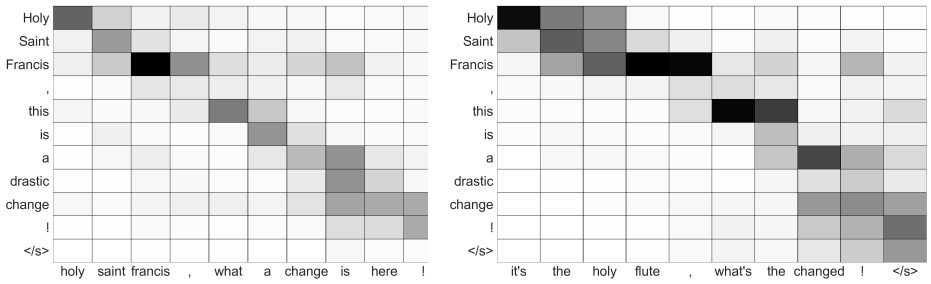



Figure 5.2: Attention matrices from a *Copy* (left) and a *simple S2S* (right) model respectively on the input sentence *"Holy Saint Francis, this is a drastic change!"* . $< s >$ and $< /s >$ are start and stop characters. Darker cells are higher-valued.

## 5.9 Related Work

There has been a small amount of prior work on style adaptation. Xu et al. [185] use phrase table based statistical machine translation to transform text to target style. In contrast, our method is an end-to-end trainable neural network. Saha Roy et al. [153] leverage different language models based on geolocation and occupation to align a text to specific style. However, their work is limited to addition of adjectives and adverbs. Our method can handle more generic transformations including addition and deletion of words.

Pointer networks [178] allow the use of input-side words directly as output in a neural S2S model, and have been used for tasks like extractive summarization [155] [190] and question answering [180]. However, pointer networks cannot generate words not present in the input. A mixture model of recurrent neural network and pointer network has been shown to achieve good performance on language modeling task [113].

S2S neural models, first devised by [169], and enhanced with a attention mechanism by [5], have yielded state-of-the-art results for machine translation (MT), summarization [151], etc. In the context of MT, various settings such as multi-source MT [193] and MT with external information [156] have been explored. Distinct from all of these, our chapter attempts to solve a Modern English $\rightarrow$ Shakespearean English style transformation task. Although closely related to both paraphrasing and MT, our task has some differentiating characteristics such as considerable source-target overlap in vocabulary and grammar (unlike MT), and different source and target language (unlike paraphrasing). [49] have devised a neural sequence-to-sequence solution for generating a portmanteau (see Chapter 2 for more) given two English root-words. Though their

task also involves large overlap in target and input, they do not employ any special copying mechanism. Unlike text simplification and summarization, our task does not involve shortening content length.

## 5.10 Conclusion

In this chapter, we studied the Constrained Creative NLG setting of diachronic style transfer from Modern English to Early Modern English, specifically, the English characteristic of Shakespeare's writing.

The dataset we had contained $\approx$ 15K parallel pairs, an order of magnitude less than the typical counts of parallel data used to train strong machine translation models [5]. Thus, we had the poor Data Availability that is a marked characteristic of this Constrained Creative class of settings being present.

We first hypothesized an underlying Creative Story based on an idealized sketch of how a human speaker would perform the task (see §5.1.1 for more). Based on this creative story, we devised Interventions to the typical E2EN2PP pipeline.

Specifically, in this chapter, our recommended approach lead to two major changes in the typical E2EN2PP pipeline. First, we recommended having a shared representation (embedding) space for source and target words initially, and pretrained this using a combined corpus of sentences from either of the sides. Third, we devised a mechanism based on retrofitting [39] to incorporate pairwise lexical source word $\rightarrow$ target word constraints from a dictionary $L_{pairwise}$, into this initial shared representation. This exploits the property of a *shared language* between source and target sides, a property likely to be shared by a large number of style transfer tasks. Second, we recommended using a mixture model of pointing/copying from input words and generating from the vocabulary to transform Modern English text to Shakespearean style English.

We demonstrated the effectiveness of our devised approach over the baselines. Our experiments revealed the utility of incorporating input-copying mechanism, and using dictionary constraints for problems with shared (but non-identical) source-target sides and sparse parallel data. We released our code publicly to foster further research on stylistic transformations on text. [11] The work has from release until August 2022 (the time of writing) been positively received by the community with over 140 citations and 46 publications explicitly using this as a baseline.

Sparknotes also provides similar translations to Modern English for the Anglo-Saxon Age epic

[11] https://github.com/harsh19/Shakespearizing-Modern-English

*Beowulf* [12] (Old English) and the famous monastic devotional chronicle *Canterbury Tales* (Middle English). A possible future direction to test the model for styles other than Shakespearean English would be to develop models to translate to these styles/languages. An additional extension would be to develop a single unified encoder-decoder model for translation to any diachronic English style, in the manner of [76], who developed an any-source language to any-target language unified translation model.

[12] tinyurl.com/d5ntme7

# Part II

# Knowledge Deficient Settings

July 24,2022

# Chapter 6

# VisCTG: Improving Plausibility for Commongen Through Retrieve-Caption-Generate (AAAI 2022)

> We're already able to see isolated cases where Cyc is learning things on its own. Some of the things it learns reflect the incompleteness of its knowledge and are just funny. For example, Cyc at one point concluded that everyone born before 1900 was famous, because all the people that it knew about and who lived in earlier times were famous people.
>
> Doug Lenat, speaking to The Austin Chronicle in 1999, about his commonsense KB/reasoner Cyc

In this chapter, we study the Commongen [99] setting, where the CG is to generate a sentence constructing a commonsense plausible situation from a given set of input concepts. Note that the CG is in some sense "underspecified" since it does not additionally provide any information e.g., through a knowledge graph or other symbolic means to ground this notion of commonsense

plausibility.

As an example, consider the input *{horse, carriage, draw}*. Two potential adequate outputs for this would be *The carriage is drawn by the horse.* and *The Sun God's carriage has seven horses drawing it.* (See Table 6.4 for more actual examples from our corpus).

We identify several critical issues in baseline model outputs for this task, like poor commonsense plausibility, inadequate pairwise lexical relationships, incomplete or missing arguments and referring expressions, and dullness/lack of specificity. This points to the scale of training data being insufficient to acquire the notion of commonsense plausibility *tabula rasa* i.e., in an inductive fashion. Furthermore, as discussed, the explicit CG is underspecified otherwise and only specifies a general requirement on the output situation of being commonsense-plausible with respect to the input concepts.

We build from a fundamental concern as a starting point to posit our ameliorating approach. Is language as a modality itself sufficient to learn the type of commonsensical pairwise lexical relationships necessary to microplan more adequate, situation-describing sentences given the input concept set? If not, why so? And more importantly, if not, how can one address this? Would incorporating information from another modality help?

The Zipfian nature of language by itself leads to a large number of concepts with relatively fewer occurrences. For pairwise co-occurrences , this problem compounds even further. However, a second, more important concern is the well noted phenomenon of reporting bias [55] — wherein unusual, exceptional and "newsworthy" events, concepts and relationships e.g., bananas being red, are mentioned more in text corpora compared to their usual, mundane and obvious counterparts e.g., bananas being yellow. Note that *reporting bias* here does not refer to any kind of *inductive bias* w.r.t. models, but the bias that exists in the medium of text itself in terms of the distribution of real world events about which text is created i.e. they are "reported" in text.

A third concern is the effect of the Gricean Maxim of Quantity, whereby speakers say only as much as is necessary, omitting information which the listener is assumed to know from commonsense. As a result, typical sentences in corpora often omit information about sufficiently obvious relationships between concepts e.g., plates being atop a table, or boats typically being afloat on an underlying water body.

We posit that these issues could indeed have a significant effect on the NLG model's learning for Commongen, and they indeed make commonsense plausibility with respect to the input as required by the CG a complex aspect to satisfy, which is characteristic of the class of Knowledge Deficient settings we're studying in this Part.

We posit that incorporating information from another modality such as vision could signifi-

cantly help dampen this effect and serve effectively as an External Knowledge Resource.

Specifically, we devise and investigate an Intervention that exploits multimodal information contained in images as an effective method for enhancing the commonsense of large, pretrained models for text generation. We perform experiments using BART and T5 as base NLG models. Note, however, that our method is agnostic to the nature of pretrained base architectures used, and does not exploit any particulars of transformers, masked pretraining or any other specifics of the BART and T5 architectures. We acknowledge the presence of a vast diversity of base architectures for generation in the research prior to these both, and use these two simply as a first step in demonstrating our approach's efficacy. We call our approach *VisCTG: Visually Grounded Concept-to-Text Generation*. VisCTG involves captioning images representing appropriate everyday scenarios, and using these captions to enrich and steer the generation process.

Specifically, the **Intervention** we devise to the E2EN2PP is the addition of an *Input Expansion Layer* between the *Input* and the *Embedding Layer*. Before passing the input string to the *Embedding Layer*, the *Input Expansion Layer symbolically augments* it with the captions of retrieved relevant images described above. Figure 6.1 illustrates our Intervention.

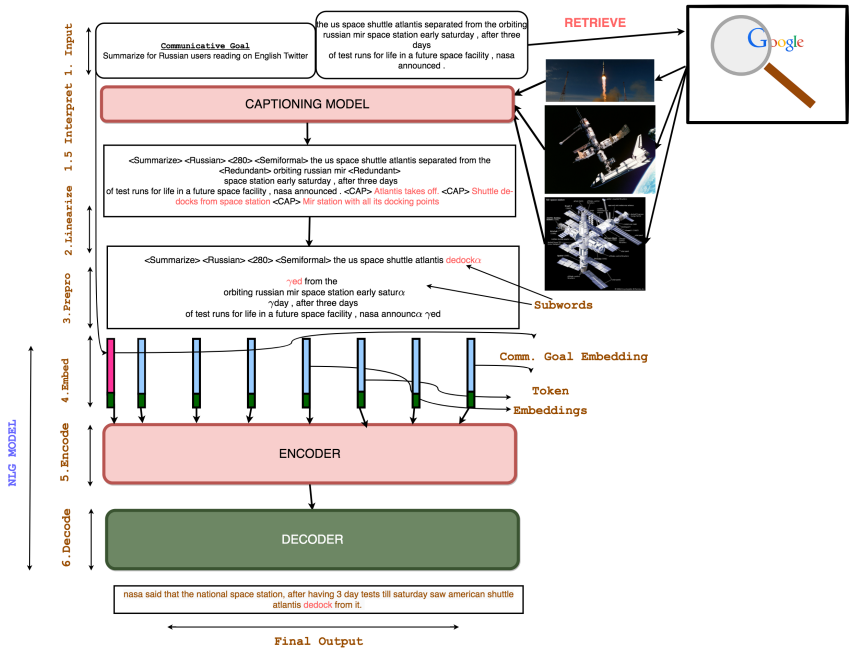



Figure 6.1: An illustration of how the E2EN2PP fleshed out in Figure 1.6 would work in action for the actual generation task and input example, after incorporating the Intervention in Chapter 6. Here, the task is to summarize the given input news article to within 280 characters. The text marked out in carrot-red in the *Final Output* , i.e *dedocked* is clearly picked up by the model from the caption-expanded portion of the input (also marked in carrot-red)

Comprehensive evaluation and analysis demonstrate that VisCTG noticeably improves model performance while successfully addressing aforementioned issues noticed in the baseline generations.

## 6.1 Introduction

Large pretrained neural models have seen increasing popularity for NLP tasks and applications. This includes SOTA text generation models such as BART [95] and T5 [138]. Larger corpora and better pretraining losses are major reasons driving these gains. However, despite increasing attention on the commonsense of models through works like COMET [13], studies have shown that even large pretrained models still struggle with commonsense tasks that humans can reason through very easily [171]. We believe that there is commonsense information in other modalities like vision, beyond what is reported [55] in text, which can possibly augment commonsense and enhance decision-making processes of text generation models.

In this chapter, we show this is true by improving the performance of Transformer-based text generation models on concept-to-text generation using visual grounding, which we call *VisCTG: Visually Grounded Concept-to-Text Generation*. Concept-to-text generation is a high-level formulation of several constrained text generation and data-to-text natural language generation (NLG) tasks. These are challenging tasks that have seen increasing interest, and involve generating natural language outputs given certain pre-conditions, e.g., specific words in the outputs, or from a collection of structured or semi-structured inputs. They typically involve converting a set of inputs into natural language text. These inputs can normally be thought of as *concepts*, or high-level words or structures, that play an important role in the generated text.

Commongen [99] involves generating sentences that effectively describe everyday scenarios from concepts sets, which are words that must appear in the output. Commongen is challenging as effective relational reasoning ability using commonsense knowledge is required. Models must also possess the compositional generalization capabilities to coalesce together different concepts. Commongen is an effective benchmark for constrained text generation and commonsense as its task formulation and evaluation methodology are rather broadly applicable.

We experiment on Commongen using BART and T5. An initial analysis (§6.3.1) of baseline generations shows several issues related to commonsense, specificity, and fluency. We hypothesize that these can be addressed through image captions (§6.3.2). Images representing everyday scenarios are commonplace, and typically logical and grounded in commonsense. Captioning models can also normally produce decent captions for everyday images, which can be used to

| | |
|---|---|
| *{stand, hold, umbrella, street}* | *{food, eat, hand, bird}* |
|  |  |
| **baseline:** A holds an umbrella while standing on the street<br>**capt:** a woman walking down a street holding an umbrella<br>**VisCTG:** A woman stands on a street holding an umbrella. | **baseline:** hand of a bird eating food<br>**capt:** a person holding a small bird in their hand<br>**VisCTG:** A bird eats food from a hand. |
| *{cat, bed, pet, lay}* | *{fence, jump, horse, rider}* |
|  |  |
| **baseline:** A cat is laying on a bed and petting it.<br>**capt:** a cat laying on a bed with a stuffed animal<br>**VisCTG:** A cat laying on a bed being petted. | **baseline:** A rider jumps over a fence.<br>**capt:** a horse is jumping over a wooden fence<br>**VisCTG:** A rider jumps a fence on a horse. |

Table 6.1: Examples of retrieved images, associated captions, baseline and VisCTG (our visually grounded model's) generations for select concept sets. Note that the images and captions are used as an intermediary to guide the final generation and thus the final generation need not be faithful to them. E.g. there is nobody petting the cat in the image or caption, but since the VisCTG output is conditioned on both the concept set and the caption, it includes *being petted*.

guide and enhance the generation process. See Table 6.1 for examples.

We use a pretrained image captioning model on MSCOCO captions [102] to caption the top retrieved images for each concept set (§6.4.1,6.4.2). We use these captions as additional information to augment inputs to our generation models (§6.4.3). Extensive evaluation (§6.6) demonstrates that VisCTG improves model performance and commonsense while addressing the baseline inadequacies.

## 6.2 Dataset, Models, and Metrics

### 6.2.1 Commongen Dataset

The original Commongen dataset [99] is made up of 35,141 concept sets (consisting of 3 to 5 keywords each) and 79,051 sentences, split into train, dev, and test splits. Since the original test

| **Dataset Stats** | $\text{Train}_{CG}$ | $\text{Dev}_{O}$ | $\text{Test}_{O}$ | $\text{Dev}_{CG}$ | $\text{Test}_{CG}$ |
|---|---|---|---|---|---|
| # concept sets | 32,651 | 993 | 1,497 | 240 | 360 |
| size = 3 | 25,020 | 493 | - | 120 | - |
| size = 4 | 4,240 | 250 | 747 | 60 | 180 |
| size = 5 | 3,391 | 250 | 750 | 60 | 180 |

Table 6.2: Statistics of Commongen dataset splits.

set is hidden, we partition the original dev set into new dev and test splits for the majority of our experiments. We do, however, ask the Commongen authors to evaluate our best VisCTG models on the original test set (more in §6.6). The training set remains the same. We refer to the original dev and test sets as $\text{dev}_{O}$ and $\text{test}_{O}$, and these new splits as $\text{train}_{CG}$, $\text{dev}_{CG}$, and $\text{test}_{CG}$. Table 6.2 contains information about these splits. Their relative sizes and distribution of concept set sizes within each are kept similar to the originals.

### 6.2.2 Models: T5 and BART

We use pretrained text generation models T5 and BART, both the base and large versions. Both are seq2seq Transformer models. T5 has strong multitask pretraining. BART is pretrained as a denoising autoencoder to reproduce original from noised text. We use their HuggingFace implementations.

We train two seeded versions of each model on $\text{train}_{CG}$ and evaluate their performance on $\text{dev}_{O}$. These serve as the baselines for our experiments. Using the numbers in Lin et al. [99] as comparison, we validate our implementations. We use the hyperparameters from Lin et al. [99], beam search for decoding, and select the final epoch as the one reaching maximum ROUGE-2 [100] on the dev split. From Table 6.3, we observe that our re-implementations reach or exceed reported results in Lin et al. [99] on most metrics.

### 6.2.3 Evaluation Metrics

We use several evaluation metrics, including those in Lin et al. [99] such as BLEU [125], CIDEr [176], SPICE [3], and coverage (cov). These (other than cov) assess similarity between human references and generations. In particular, CIDEr captures a combination of sentence similarity, grammaticality, saliency, importance, and accuracy. SPICE maps texts to semantic scene graphs and calculates an F-score over these graphs' tuples. Lin et al. [99] note that SPICE correlates

| Model\Metrics | BLEU-4 | CIDEr | SPICE |
|---|---|---|---|
| Reported BART-large | 27.50 | 14.12 | 30.00 |
| Reported T5-base | 18.00 | 9.73 | 23.40 |
| Reported T5-Large | 30.60 | 15.84 | 31.80 |
| Our BART-base | 28.30 | 15.07 | 30.35 |
| Our BART-large | **30.20** | **15.72** | **31.20** |
| Our T5-base | **31.00** | **16.37** | **32.05** |
| Our T5-large | **33.60** | **17.02** | **33.45** |

Table 6.3: Comparing $dev_O$ performance of our re-implemented models to those in Lin et al. [99]. Bold represents where we reach/exceed reported numbers. Results averaged over two seeds for our models. Lin et al. [99] did not report BART-base. See §6.2.3 for metric explanationsfor comparison of all metrics.

highest with human judgment for Commongen. Cov measures the average percentage of input concepts covered by the output text in any form.

We also use BERTScore [191] and Perplexity (PPL). BERTScore measures BERT [32] embeddings similarity between individual tokens, serving as a more semantic rather than surface-level similarity measure. We multiply by 100 when reporting BERTScore. PPL serves as a measure of fluency, with lower values representing higher fluency. We use GPT-2 [136] for PPL. For all metrics other than PPL, higher means better performance.

## 6.3 Initial Analysis and Motivation

### 6.3.1 Baseline Model Generations

We conduct an initial analysis of the baseline model outputs, and observe that several lack fluency and commonsense plausibility. Some are more like phrases than complete, coherent sentences, e.g. *"body of water on a raft"*. Others miss important words, e.g. *"A listening music and dancing in a dark room"* misses a noun before *listening*. A large portion of generations are generic and bland, e.g. *"Someone sits and listens to someone talk"*. This may be an instance of the *dull response problem* faced by generation models [35, 177], where they prefer safe and frequent responses independent of input information.

Many generations are also implausible. For example, *"body of water on a raft"* is implausible as the phrases *"body of water"* and *"a raft"* are coalesced together incorrectly i.e., the roles of *raft* and *body of water* are incorrectly swapped. A similar issue occurs with the *{horse, carriage, draw}* example in Table 6.4. At times the models also cannot understand what certain nouns can do i.e., their affordances e.g. *"A dog checking his phone on a pier."*. Several other examples of

| Concept Set | Baseline Generation | Human Reference |
|---|---|---|
| {horse, carriage, draw} | horse drawn in a carriage | The carriage is drawn by the horse. |
| {dog, house, eat} | A dog eats hay in a house | The dog eats food inside the house. |
| {cow, horse, lasso} | A cow is lassoing a horse. | A group of men riding horses lassoing a cow. |

Table 6.4: Example generations from our baseline models versus human references.

this can be found in Table 6.4.

### 6.3.2 Images and Captions

Images that represent everyday scenarios are quite prevalent for almost any reasonable concept set. Further, the images are typically grounded in commonsense. For example, searching *{cow, horse, lasso}* will result in many images of cowboys riding horses and lassoing cows, rather than the illogical situation of *"A cow is lassoing a horse."* described by the baseline generation in Table 6.4. Many everyday images are relatively similar to those in image captioning datasets such as MSCOCO, so pretrained captioning models should work quite effectively. We thus hypothesize that using images and their captions to visually ground concept-to-text generation can potentially deal with issues mentioned in §6.3.1. Retrieved images with corresponding captions generated by a pretrained image captioning model (see §6.4.2) and final baseline and VisCTG generations for select concept sets are in Table 6.1.

Textual corpora also suffer from *reporting bias* [55], where everyday, commonsense albeit "uninteresting" actions (walking), objects (bench) and facts (bananas are yellow) are underrepresented compared to real-world frequency, while "newsworthy" actions (murdering), objects (spaceships) and facts (blue GMO bananas) are exaggerated. This seeps even into large pretrained text models [159]. Using visual data and models dampens this bias, likely improving the commonsense of generations.

## 6.4 Methodology

### 6.4.1 Image Retrieval

We first obtain images for each concept set in our three splits. Image captioning datasets such as MSCOCO and Flickr are typically too small and focused to be effective for our purposes since we must cover numerous different concept sets. Further, a search engine is more generalizable.

We decide to use Google Images. On a sample of concept sets, the retrieved images using

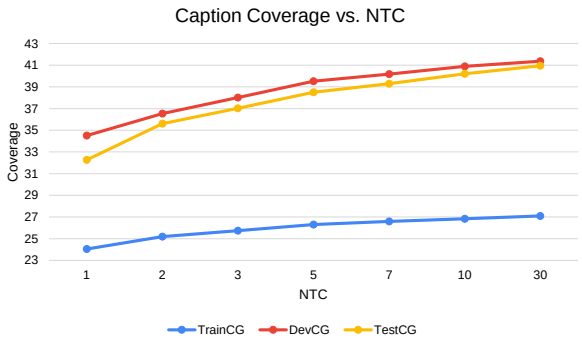



Figure 6.2: Graph displaying the average coverage (out of 100) by the top NTC captions in aggregate per concept set.

| **Augmented Input → Final Generation** |
|---|
| wave fall board surfer <s> a surfer riding a wave on a surfboard → **A surfer is falling off his board into the waves.** |
| dance stage front crowd <s> a crowd of people watching a man on a stage <s> a man is holding a microphone in front of a crowd → **A man dances in front of a crowd on stage.** |
| stand hold umbrella street <s> a woman walking down a street holding an umbrella <s> a woman walking down a street holding an umbrella <s> a girl holding a pink umbrella in a city <s> a man holding an umbrella in a city <s> a group of people standing under a umbrella → **A group of people standing on a street holding umbrellas.** |

Table 6.5: Examples of augmented inputs and final generations for varying values of NTC.

other search engines were inappropriate; they did not incorporate most input keywords nor handle homonyms well. For example, *"sports+fan+watch"* yields images of fans watching a sports game on Google images, but images of hand watches on Bing and DuckDuckGo.

We queried input concept sets by concatenating keywords with plus signs (+), and used *simple-image-scraper*[1] to obtain URLs of the top 30 results. The image was scraped only if the URL ended in *.png*, *.jpeg*, *.jpg*, or *.gif*. The received content was verified to be valid images using *pillow*[2], otherwise skipped. Retrieved images were typically of high quality and corresponded well to the concepts. See Table 6.1 for examples.

[1] https://pypi.org/project/simple-image-download/
[2] https://pypi.org/project/Pillow/

### 6.4.2 Image Captioning

After retrieving images, we use a PyTorch-based implementation[3] of the FC image captioning model [107, 147], which generates a caption via an LSTM initialized with a pseudo token obtained by feeding the image into a deep CNN followed by a linear projection. We use a pretrained FC model trained on the MSCOCO dataset with pretrained Resnet-101 image features. As most of our retrieved images represent everyday scenarios and are relatively similar to those in MSCOCO, the pretrained model performs quite well. See example captions in Table 6.1.

### 6.4.3 Caption Selection and Input Augmentation

After we have captions $S_c = \{c_1, c_2, ..., c_n\}$ for each concept set in all three splits, we reorder them by descending coverage to the concept set to obtain $S_{c'} = \{c'_1, c'_2, ..., c'_n\}$. If two captions are tied for coverage, we keep them in their original search result order. This allows us to select the captions that have highest coverage and are most relevant.

Since most retrieved images and corresponding captions cover only a fraction of the entire concept set, and the quality of each varies, we hypothesize that using multiple captions for generation may lead to more robust and higher-quality outputs with more coverage. The models may learn to assimilate together information from caption(s) while generating final texts. Hence, we try experiments using different numbers of top captions within $S_{c'}$, a parameter we call $NTC$ (Number of Top Captions). We try $NTC = 1, 2, 3, 5, 7, 10$, and do not go above $NTC = 10$ as Figure 6.2 shows that coverage gains from $10 \rightarrow 30$ are minor. Figure 6.2 also illustrates that captions have relatively low individual coverage, especially compared with outputs from models trained on Commongen, which is why we do not use them as a baseline.

The captions are concatenated together and onto the concept set using <s> separator tokens. These serve as augmented inputs to BART and T5. They learn to convert these augmented inputs to human references during training, and are fed the augmented inputs (corresponding to the value of NTC) during validation and testing. Some examples of augmented inputs and generations can be found in Table 6.5.

[3] https://github.com/ruotianluo/self-critical.pytorch

## 6.5 Experiments

### 6.5.1 Model Training and Selection

For training VisCTG models, we mainly follow baseline hyperparameters, barring learning rate (LR) that is tuned per NTC value, and the maximum encoder length which is chosen depending on the tokenizer and value of NTC to ensure the entire input sequence can fit onto the encoder. We train two seeds per model.

For each model, we choose the epoch corresponding to highest ROUGE-2 on $\text{dev}_{CG}$, and use beam search for decoding. NTC itself is a hyperparameter, so while we train separate versions of each model corresponding to different NTC values, the final chosen models correspond to the NTC values that performed best on $\text{dev}_{CG}$ when averaged over both seeds. We then use the final chosen models to generate on both $\text{test}_{CG}$ and $\text{test}_{O}$, and report the results in §6.6.

### 6.5.2 Human Evaluation

We conduct two human evaluations: one using Amazon Mechanical Turk (AMT), and one using an expert linguist. For the AMT study, we ask annotators to evaluate 86 $\text{test}_{CG}$ examples per model. Our evaluation is based on pairwise comparison of VisCTG and baseline model outputs. We ask human annotators to choose which amongst the two outputs (presented in a random order per example) has better *Overall Quality*. There are 3 choices: O1: VisCTG is better, O2: baseline is better, O3: both are indistinguishable. To aggregate multiple annotations per example, we find the fraction of responses towards each outcome value as the per-example distribution. We then find the sample mean of this outcome distribution over all examples. For sample mean and significance testing, we are interested in the values for O1 vs. O2.

For the expert linguist study, our expert is a native English speaker with a graduate degree in linguistics from a North American university. The expert is asked to annotate three aspects for 50 BART-large[4] $\text{test}_{CG}$ examples: *Overall Quality (Overall)*, *Commonsense Plausibility (Commonsense)*, and *Fluency (Fluency)*. For all aspects, we have a pairwise-comparison evaluation setup similar to that for AMT.

| | **BART-base** ($NTC = 5$) | | | **BART-large** ($NTC = 2$) | | |
|---|---|---|---|---|---|---|
| **Metrics** | Baseline | VisCTG | p-value | Baseline | VisCTG | p-value |
| ROUGE-1 | 43.96±0.03 | **45.44**±0.08 | 1.58E-05 | 45.67±0.25 | **46.91**±0.31 | 1.58E-05 |
| ROUGE-2 | 17.31±0.02 | **19.15**±0.21 | 1.58E-05 | 18.77±0.04 | **20.36**±0.05 | 1.58E-05 |
| ROUGE-L | 36.65±0.00 | **38.43**±0.07 | 1.58E-05 | 37.83±0.29 | **39.23**±0.01 | 1.58E-05 |
| BLEU-1 | 73.20±0.28 | **75.65**±0.78 | 6.94E-05 | 74.45±0.21 | **78.80**±0.28 | 6.94E-05 |
| BLEU-2 | 54.50±0.14 | **59.05**±0.07 | 6.94E-05 | 56.25±0.78 | **61.60**±0.85 | 6.94E-05 |
| BLEU-3 | 40.40±0.14 | **44.90**±0.42 | 6.94E-05 | 42.15±0.49 | **47.00**±0.71 | 6.94E-05 |
| BLEU-4 | 30.10±0.14 | **34.10**±0.57 | 3.82E-03 | 32.10±0.42 | **36.25**±0.78 | 2.08E-04 |
| METEOR | 30.35±0.35 | **31.95**±0.07 | 6.94E-05 | 31.70±0.14 | **34.00**±0.14 | 6.94E-05 |
| CIDEr | 15.56±0.10 | **16.84**±0.05 | 6.94E-05 | 16.42±0.09 | **18.35**±0.13 | 6.94E-05 |
| SPICE | 30.05±0.07 | **31.80**±0.28 | 6.94E-05 | 31.85±0.21 | **34.60**±0.28 | 6.94E-05 |
| BERTScore | 59.19±0.32 | **61.44**±0.02 | 1.58E-05 | 59.95±0.29 | **62.85**±0.30 | 1.58E-05 |
| Coverage | 90.43±0.17 | **90.66**±1.39 | 0.33* | 94.49±0.53 | **96.49**±0.24 | 1.58E-05 |
| PPL | 80.39±3.65 | **72.45**±0.79 | 1.58E-05 | 80.37±4.51 | **68.46**±5.90 | 1.58E-05 |

Table 6.6: Automatic eval results for BART on test$_{CG}$ over two seeds. Bold corresponds to best performance on that metric. We include stat sig p-values (from Pitman's permutation test [133]) for VisCTG compared to the baseline. Insignificant ones ($\alpha = 0.1$) marked with *.

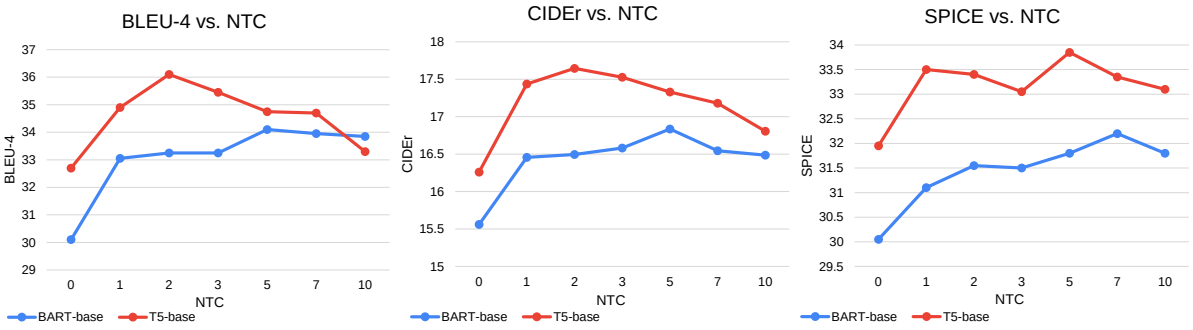



Figure 6.3: BLEU-4, CIDEr, and SPICE on test$_{CG}$ over different values of NTC for BART-base and T5-base.

## 6.6 Results and Analysis

Automatic evaluation results on test$_{CG}$ are in Tables 6.6 and 6.7, and results on test$_{O}$ in Table 6.8.[5] Graphs displaying BLEU-4, CIDEr, and SPICE (the metrics on the Commongen leaderboard[6]) on test$_{CG}$ over different NTC values are in Figure 6.3. Human evaluation results on test$_{CG}$ are in Tables 6.9 and 6.10. Optimal NTC values for BART-base, BART-large, T5-base, and T5-large are 5, 2, 2, and 1, respectively. These are the VisCTG results reported in the aforementioned tables.

[4]Since this is the best performing VisCTG model - see §6.6.
[5]Evaluated by the Commongen authors on their hidden test set.
[6]https://inklab.usc.edu/Commongen/leaderboard.html

| | **T5-base** ($NTC = 2$) | | | **T5-large** ($NTC = 1$) | | |
|---|---|---|---|---|---|---|
| **Metrics** | Baseline | VisCTG | p-values | Baseline | VisCTG | p-values |
| ROUGE-1 | 44.63±0.13 | **46.26**±0.07 | 1.58E-05 | 46.32±0.26 | **46.93**±0.22 | 7.26E-04 |
| ROUGE-2 | 18.40±0.14 | **19.78**±0.30 | 1.58E-05 | 19.59±0.12 | **20.01**±0.23 | 0.02 |
| ROUGE-L | 37.60±0.16 | **38.91**±0.27 | 1.58E-05 | 39.20±0.21 | **39.52**±0.43 | 0.06 |
| BLEU-1 | 73.60±0.85 | **76.80**±0.28 | 6.94E-05 | 77.55±0.35 | **78.65**±0.21 | 4.65E-03 |
| BLEU-2 | 57.00±0.71 | **60.30**±0.28 | 6.94E-05 | 60.80±0.28 | **61.55**±0.35 | 0.07 |
| BLEU-3 | 42.75±0.49 | **46.25**±0.64 | 6.94E-05 | 46.50±0.00 | **47.10**±0.57 | 0.11* |
| BLEU-4 | 32.70±0.42 | **36.10**±0.85 | 6.94E-05 | 36.20±0.14 | **36.40**±0.28 | 0.21* |
| METEOR | 31.05±0.49 | **32.70**±0.00 | 6.94E-05 | 33.20±0.00 | **33.65**±0.49 | 0.49* |
| CIDEr | 16.26±0.25 | **17.65**±0.02 | 6.94E-05 | 17.79±0.01 | **17.94**±0.25 | 0.23* |
| SPICE | 31.95±0.07 | **33.40**±0.28 | 6.94E-05 | 33.90±0.42 | **34.55**±0.21 | 0.03 |
| BERTScore | 61.40±0.34 | **62.42**±0.17 | 1.58E-05 | 62.67±0.09 | **62.72**±0.03 | 0.34* |
| Coverage | 90.96±1.77 | **94.48**±1.39 | 1.58E-05 | 94.40±0.02 | **95.95**±0.45 | 1.58E-05 |
| PPL | 83.04±1.62 | **77.50**±3.86 | 3.16E-05 | 81.78±4.63 | **73.41**±4.32 | 1.58E-05 |

Table 6.7: Automatic eval results for T5 on test$_{CG}$ over two seeds. Bold corresponds to best performance on that metric. We include stat sig p-values (from Pitman's permutation test [133]) for VisCTG compared to the baseline. Insignificant ones ($\alpha = 0.1$) marked with *.

Table 6.11 contains qualitative examples.

### 6.6.1 Analysis of Automatic Evaluation Results

We see from Tables 6.6 and 6.7 that VisCTG outperforms the baselines on all metrics across the models on test$_{CG}$. Performance gains are strong and statistically significant for BART-base, BART-large, and T5-base. VisCTG appears relatively less effective for T5-large which is the strongest baseline, and hence improving its performance may be more difficult.

From Table 6.8, we see that VisCTG models substantially outperform corresponding baselines reported in Lin et al. [99] on test$_O$. T5-base VisCTG outperforms the reported T5-base and large baselines across metrics, and BART-base VisCTG performs similarly to the reported BART-large baseline. BART-large VisCTG outperforms the reported baseline, EKI-BART [38], and KG-BART [104]. These are SOTA published Commongen BART models that use external knowledge from corpora and KGs. We show that visual grounding is more effective, and BART-large VisCTG would place very high on the leaderboard.[6] T5-large VisCTG outperforms the reported baseline, but lags behind the SOTA published RE-T5 [179].

Figure 6.3 shows that as NTC increases, BLEU-4, CIDEr, and SPICE increase to a peak, and taper off after. This is expected as we saw in Figure 6.2 that the rate of increase of coverage declines with larger NTC. The latter images and captions are of diminishing quality, and hence

| **Models\Metrics** | ROUGE-2/L | | **BLEU-3/4** | | METEOR | **CIDEr** | **SPICE** | Coverage |
|---|---|---|---|---|---|---|---|---|
| T5-base (reported baseline) | 14.63 | 34.56 | 28.76 | 18.54 | 23.94 | 9.40 | 19.87 | 76.67 |
| T5-large (reported baseline) | 21.74 | 42.75 | 43.01 | 31.96 | 31.12 | 15.13 | 28.86 | 95.29 |
| BART-large (reported baseline) | 22.02 | 41.78 | 39.52 | 29.01 | 31.83 | 13.98 | 28.00 | 97.35 |
| EKI-BART [38] | - | - | - | 35.945 | - | 16.999 | 29.583 | - |
| KG-BART [104] | - | - | - | 33.867 | - | 16.927 | 29.634 | - |
| RE-T5 [179] | - | - | - | **40.863** | - | **17.663** | **31.079** | - |
| T5-base VisCTG | 22.83 | 44.98 | 45.749 | **34.722** | 31.809 | **16.173** | **28.808** | 92.92 |
| T5-large VisCTG | 23.83 | 45.76 | 47.376 | 36.409 | 33.012 | 16.815 | 29.629 | 95.54 |
| BART-base VisCTG | 21.73 | 43.43 | 43.235 | **32.291** | 30.86 | **15.187** | **27.403** | 88.98 |
| BART-large VisCTG | 23.68 | 45.07 | 48.031 | **36.939** | 33.215 | **17.199** | **29.973** | 94.86 |

Table 6.8: Automatic eval results of VisCTG models on test$_O$, evaluated by Commongen authors. We compare to reported baseline numbers in Lin et al. [99] (they did not evaluate BART-base), and models on their leaderboard with publications at time of writing that outperform baselines. Their leaderboard reports BLEU-4, CIDEr, and SPICE. Bold corresponds to best performance (for those three) per model type+size.

| Model | **O1** | **O2** | **O3** | **IAA** |
|---|---|---|---|---|
| **BART-base** | **0.45** | 0.33 | 0.22 | 0.72 |
| **BART-large** | **0.62** | 0.18 | 0.20 | 0.55 |
| **T5-base** | **0.46** | 0.33 | 0.21 | 0.72 |
| **T5-large** | **0.46** | 0.34 | 0.20 | 0.74 |

Table 6.9: Avg. AMT eval results on test$_{CG}$ for *overall quality*. O1: VisCTG wins, O2: baseline wins, O3: both indistinguishable. Bold corresponds to higher fractional outcome between O1 and O2. All results are statistically significant based on paired two-tailed t-tests and $\alpha = 0.1$. The inter-annotator agreement (IAA) is the average direct fractional agreement (where both annotators choose O1 or O2) over all examples. See §6.5.2 for further details.

| Model | Aspect | **O1** | **O2** | **O3** |
|---|---|---|---|---|
| **BART-large** | Overall | **0.44** | 0.24 | 0.32 |
| | Commonsense | **0.32** | 0 | 0.68 |
| | Fluency | **0.56** | 0.12 | 0.32 |

Table 6.10: Avg. expert linguist eval results on test$_{CG}$ for BART-large. O1: VisCTG wins, O2: baseline wins, O3: both indistinguishable. Bold corresponds to higher fractional outcome between O1 and O2 per aspect. See §6.5.2 for further details.

using too many negatively affects model performance.

### 6.6.2 Analysis of Human Evaluation Results

Table 6.9 shows that VisCTG outperforms the baseline on all four models based on human annotators (with high IAA). Annotators, on average, prefer VisCTG outputs over baseline outputs

on overall quality, especially for BART-large. Table 6.10 illustrates that VisCTG outperforms the baseline model for BART-large based on an expert linguist's perspective. VisCTG outputs are highly preferred, on average, over the baseline on all three aspects of overall quality, commonsense, and fluency. This aligns with our automatic results in §6.6.1, where VisCTG outperforms the baselines across all models.

### 6.6.3 Qualitative Analysis

Table 6.11 shows several baseline outputs that contain issues from §6.3.1, e.g., incomplete and/or illogical sentences. Human references are all fluent and logical. VisCTG can usually generate much higher-quality text than the baselines.

The baseline outputs for ex. 1-2 are phrases lacking arguments, and are illogical for ex. 1-3. Using captions, VisCTG successfully adjusts semantic roles of entities, replaces incorrect subjects, fixes dependency structure, and grounds generations in commonsense. For ex. 1, captions are of the form "{X} *sitting on a chair with* {Y}", where {X} is a subject and {Y} an object. VisCTG output has similar structure, being fluent and logical with higher coverage. The baseline output also has an incorrect subject of *"hands"*. Our VisCTG output contains an additional entity (not present in the input set) of *"boy"* as subject, likely since it is a subject in the captions. This highlights the usefulness of visual grounding, as the image space can provide additional commonsense information not present in the text (e.g. toys are associated with children/boys). For ex. 2, the baseline output treats *"hand of a bird"* as a single entity, the subject. Captions separate *"bird"* and *"hand"* into two, likely guiding the VisCTG output to do so. For ex. 3, the baseline misplaces *"bus"* as subject. Captions are of form "{X} *sitting on a bench* {Y}", where {X} is a logical subject and {Y} is an expression. The VisCTG output has this structure, with correct subject and commonsense, and higher coverage. Overall, we see that visual grounding guides the model to learn which nouns/subjects can perform which actions (e.g. *"hands"* cannot sit on a chair but a *"boy"* can), which is a major baseline deficiency discussed in §6.3.1.

For ex. 4, the baseline output lacks a subject that the captions contain, likely guiding the VisCTG output to contain one: *"a man"*. For ex. 5, the baseline output is generic due to uses of *"someone"*. VisCTG's output is more specific and refers to *"man"*, likely because the caption (though not very fitting) includes a *"man"* subject. Even for captions that fit the concepts less, structure and fluency can still be exploited.

Overall, we see that the baselines simply try to coalesce together the input concepts into a form of English syntax, often failing to do so effectively. VisCTG models can produce more

grammatical, fluent, and logical text by exploiting the syntactic and dependency structures of the captions. Further, the visual grounding improves the commonsense of the generations. The images inherently capture commonsense by representing everyday scenarios, and this commonsense info is rarely explicitly included in text. Hence, large text-based models such as our baselines tend to not know this info, whereas VisCTG models learn it through the grounding.

VisCTG is, however, still a far way off from perfect. For ex. 6, its output is less logical and lower coverage than the baseline's. The captions are all simplistic and low coverage; the first is illogical, and some others are of the form *"a bunch of apples {...} on a tree"*, likely negatively impacting the generation. Ex. 4's human reference is creative, which is an area where VisCTG still lacks in comparison. For ex. 5, while VisCTG edits *"someone"* to *"man"*, it is unable to merge the two instances of *"man"* or adjust the sentence to be more coherent. These weaknesses are likely because captions tend to be simplistic (due to the captioning model's training data), limiting VisCTG's ability to make heavier edits. VisCTG, unsurprisingly, appears to depend quite heavily on the captions, and hence the quality of the images and captioning model.

## 6.7 Related Work

**Constrained Text Generation:** There have been several works on constrained text generation. Miao et al. [114] use Metropolis-Hastings sampling to determine Levenshtein edits per generation step. Feng et al. [42] devise Semantic Text Exchange to adjust topic-level text semantics.

**Data-to-text NLG:** E2E-NLG [36] and WebNLG [52] are two popular NLG benchmarks with structured inputs - meaning representation (MR) and triple sequences, respectively. Montella et al. [116] use Wiki sentences with parsed OpenIE triples as weak supervision for WebNLG.

**Commonsense Injection and Incorporation:** One large commonsense knowledge graph (KG) is COMET, trained on KG edges to learn connections between words and phrases. EKI-BART [38] and KG-BART [104] use external knowledge (from corpora and KGs) to improve BART's performance on Commongen. Distinctly, VisCTG uses visual grounding and shows higher performance (see §6.6). Visual Commonsense Reasoning (VCR) [188] involves answering commonsense-related multiple-choice questions about images. Our work uniquely focuses on injecting commonsense into seq2seq Transformer models like BART and T5 for text generation.

**Multimodal Machine Learning and NLP:** There has been more work on multimodality, in areas like representation and video captioning, but little for constrained and data-to-text NLG [6, 51]. There is work on pretrained multimodal models like ViLBERT [106], which are mainly encoders that jointly represent images and text rather than seq2seq models, and would be ill-suited for generation. Further, unlike these models which are pretrained, VisCTG exploits per-example visual information to fix specific issues for each concept set.

## 6.8 Conclusion and Future Work

In conclusion, we motivated and explored the use of visual grounding as an External Knowledge Resource for improving the abilities of Transformer models at the Commongen generation setting. We christen our method VisCTG: Visually Grounded Concept-to-Text Generation. Extensive experiments on BART and T5 showed the efficacy of our devised Intervention and the resultant VisCTG method on the Commongen task. Comprehensive evaluation and analysis showed that VisCTG boosts model performance and commonsense while addressing baseline deficiencies observed as a byproduct of a) The underspecified (explicit) CG and b) Training data insufficient to facilitate knowledge acquisition tabula rasa.

Our empirical findings support the hypothesis that relying on language and the CG alone is insufficent to learn a good NLG model for Commongen and could cause the issues we observed in our baseline outputs e.g., those in Table 6.4. They also confirm our intuition that incorporating information from the visual modality as an External Knowledge Resource can in part ameliorate this insufficiency. Furthermore, they support the case for intervening in the E2ENLP and introducing an *Input Expansion Layer* between the *Input Layer* and *Embedding Layer* to symbolically augment the input with captions of retrieved images, as described in Figure 6.1.

In this chapter, we successfully devised an intervention to SOTA pretrained generator models to improve their microplanning ability, and consequently, their output quality, while accomplishing the *generative commonsense reasoning* task a.k.a *Commongen* [99] task.

Potential future work includes improving image search and captioning, e.g. better selection of images during retrieval or using a stronger captioning model. Video captioning and image generation rather than retrieval can also be explored. Further, VisCTG can be investigated for other data-to-text NLG tasks, e.g. WebNLG.

### 6.8.1 Broader Takeaways

The broader takeaway from our findings is that in any task where the communicative goal (or the "input text" part of the communicative goal) leaves gaps w.r.t. the relations amongst different parts of the input to be filled in a "plausible", commonsensical kind of way, and leaves the notion of plausibility otherwise unspecified, reporting bias is naturally bound to be a problem for any model trained on typical natural language corpora. Examples of such input information include sets of concepts (as in our case), recipes or Wikipedia infoboxes. In such situations, using another modality (which could be something like images/audio, or even another language) which has *lesser* or *different* reporting bias, to "expand" the underspecified input information is a potential architectural enhancement to explore in order to improve the microplanning i.e., the plausibility and internal structure of sentences. Implementing this kind of input "expansion" requires a mechanism to ground the input into the other modality, and then reground it back. In the case of concept-to-text generation tasks, the simple nature of the input information and the availability of well optimized search engines greatly simplifies and streamlines the grounding process, which is unlikely to be as straightforward in the general case. Furthermore, even the regrounding process is simplified due to the presence of well-developed captioning models.

As an example, consider the task of summarizing social media posts from forums/subreddits related to a religion with its primary mode of religious discourse and scripture being a non-English one e.g., Islam or Judaism, who have the bulk of their scriptures, commentaries and other resources in Arabic and Hebrew respectively, which we shall refer to henceforth as the *scriptural language*. In this case, one potential architectural enhancement based on the same principle as ours, would be as follows:

1. Translate the input social media post $x$ to the scriptural language using an off-the-shelf English→scriptural language translation model.
2. Take the translated input $T_{scriptural}(x)$ and use it to retrieve similar sentences from any large corpora of religious texts, commentary, discourse in the scriptural language.
3. Translate back each of the retrieved sentences $s^i_{candidate} \in Retrieve(T_{scriptural}(x))$ to English using an off-the-shelf scriptural language→English translation model.
4. Just as we did with the captions of retrieved images, augment the input $x$ with the set of top $K$ most relevant from amongst the back-translated candidates $T_{English}(s^i_{candidate})$.
5. Train the models with the now-augmented inputs.

Here, the scriptural language plays the same role as images/visual modality in this chapter. Since

the scriptural language is more likely to have a wider coverage and range of religious terminology, arguments and text, it would naturally suffer from lesser reporting bias. The English→scriptural language and scriptural language→English models serve as the grounding and regrounding mechanisms respectively.

The second takeway from our findings is that they underscore the added potential utility of joint spaces which embed together different modalities, such as the very recent CLIP representation [137] from OpenAI. Instead of using retrieval from a search engine as in our case, one can directly compute cross-modal similarities in this space, e.g., between a given input text and an image.

## 6.9 Appendices

## 6.10 Full Re-implementation versus Reported Model Numbers

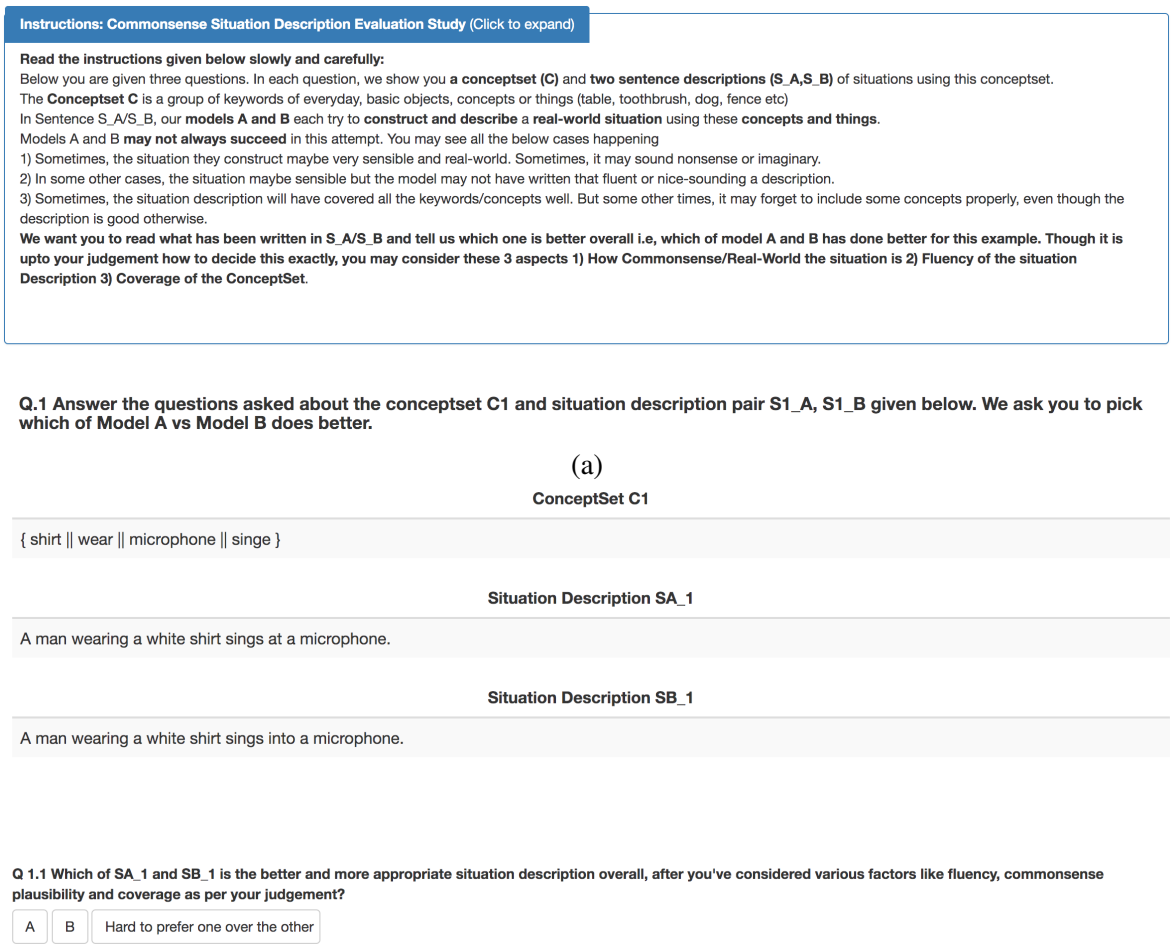
See Table 6.12 for a full comparison (across all metrics) of our re-implemented Commongen models compared to the original reported baseline numbers in Lin et al. [99].

## 6.11 Pretrained FC Image Captioning Model Details

The image encoder is a pretrained Resnet-101 [146], where the global avg. pooling of the final convolutional layer output, a vector of dim. 2048, is taken per image. Spatial features are extracted from the output of a Faster R-CNN [4, 146] with ResNet-101 [62], trained by object and attribute annotations from Visual Genome [91]. For captioning, the dimensions of LSTM hidden state, image feature embedding, and word embedding are all set to 512. Please see Luo et al. [107], particularly Sections 3.3 and 5.1, and Rennie et al. [147], particularly Sections 2 and 5, for more.

## 6.12 BART and T5 Model Training and Generation Details

T5-large has 770M params, T5-base 220M params, BART-large 406M params, and BART-base 139M params. Two seeded versions of each baseline and VisCTG model are trained. For decoding, we use beam search with a beam size of 5, decoder early stopping, a decoder length penalty of 0.6, a decoder maximum length of 32, and a decoder minimum length of 1 for all models. We use a maximum encoder length of 32 for the baselines and for the VisCTG models: up to 160 for BART and 256 for T5. A batch size of 64 for T5-base and BART-base, 32 for BART-large, and




(a)

(b)

Figure 6.4: Snapshots of human evaluation: a) instructions seen by annotator and b) an example with questions.

8 for T5-large is used for training. We 500 warmup steps for BART-large, and 400 for T5-base, T5-large, and BART-base. All models are trained up to a reasonable number of epochs (e.g. 10 or 20) and early stopping using our best judgment is conducted, e.g. if metrics continuously drop for several epochs. Learning rates for VisCTG models were determined by trying several values (e.g. from 1e-6 to 1e-4), and finding ones which result in decent convergence behavior, e.g. dev metrics increase steadily and reach a maximum after a reasonable number of epochs. For the final models (e.g. best NTC values for VisCTG), learning rates are (each set consists of {BART-base,BART-large,T5-base,T5-large}): baselines = {3e-05,3e-05,5e-05,2e-05}, VisCTG = {1e-05,5e-06,2e-05,2e-05}.

Google Colab instances were used for training, which used either a single V100 or P100 GPU. Most of the training experiments were performed using a single V100. BART-base models trained

in approx. 1 hour, T5-base models in approx. 1.5 hours, BART-large models in approx. 2 hours, and T5-large models in approx. 6 hours.

## 6.13 Human Evaluation Details

The Amazon Mechanical Turk (AMT) human evaluation was performed through paid annotators on AMT. Annotators were from Anglophone countries with $> 97\%$ approval rate. Each example was evaluated by up to three annotators. Each AMT task page or HIT contained 2 actual examples and a “quality-check" example in random order. Specific instructions and a question snippet can be seen in Figure 6.4.

On every annotation page, we include one randomly chosen “quality-check" example from a list of such hand-crafted examples, in addition to two actual examples with VisCTG and baseline outputs. The hand-crafted examples are constructed to have an obviously good and an obviously bad output pair, and are sourced from Lin et al. [99]. If an annotator answers the quality-check question wrong (e.g. they choose the obviously bad output), their two remaining actual example annotations are excluded while compiling results.

The time given for each AMT task instance or HIT was 8 minutes. Sufficient time to read the instructions, as calibrated by authors, was also considered in the maximum time limit for each HIT/task. Annotators were paid 98 cents per HIT. The rate of payment ($7.35/hour) exceeds the minimum wage rate for the USA ($7.2/hour) and hence constitutes fair pay. We neither solicit, record, request, or predict any personal information pertaining to the AMT crowdworkers.

The expert linguist evaluation included a human subject institutional board protocol and a rate of payment of $15/hour, also exceeding the minimum wage rate for the USA.

## 6.14 Further Qualitative Examples

See Table 6.13 for further qualitative examples.

| Method | Text |
|---|---|
| Concept set | {sit, chair, toy, hand} (example 1) |
| Captions | a little girl sitting on a chair with a teddy bear <s> a small child sitting on a chair with a teddy bear <s> a young boy sitting on a chair with a skateboard <s> a man sitting on a chair with a remote |
| BART-base-BL | hands sitting on a chair |
| BART-base-VisCTG | A boy sitting on a chair with a toy in his hand. |
| Human reference | A baby sits on a chair with a toy in one of its hands. |
| Concept set | {food, eat, hand, bird} (example 2) |
| Captions | a bird is perched on a branch with a hand <s> a person holding a small bird in their hand |
| BART-large-BL | hand of a bird eating food |
| BART-large-VisCTG | A bird eats food from a hand. |
| Human reference | A small bird eats food from someone's hand. |
| Concept set | {bench, bus, wait, sit} (example 3) |
| Captions | a man sitting on a bench with a book <s> a person sitting on a bench with a laptop |
| T5-base-BL | A bus sits on a bench. |
| T5-base-VisCTG | A man sits on a bench waiting for a bus. |
| Human reference | The man sat on the bench waiting for the bus. |
| Concept set | {jacket, wear, snow, walk} (example 4) |
| Captions | a young boy in a red jacket is standing in the snow <s> a man in a red jacket is standing in the snow |
| BART-large-BL | walking in the snow wearing a furry jacket |
| BART-large-VisCTG | A man is walking in the snow wearing a jacket. |
| Human reference | Jamie took a walk out into the snow with only a T shirt on and instantly went back inside to wear his jacket. |
| Concept set | {hold, hand, stand, front} (example 5) |
| Captions | a man holding a pair of scissors in front of a wall |
| T5-large-BL | Someone stands in front of someone holding a hand. |
| T5-large-VisCTG | A man stands in front of a man holding a hand. |
| Human reference | A man stands and holds his hands out in front of him. |
| Concept set | {bag, put, apple, tree, pick} (example 6) |
| Captions | a person holding a apple in a tree <s> a bunch of apples are growing on a tree <s> a close up of a green apple with a tree <s> a bunch of apples are growing on a tree |
| BART-base-BL | A man is putting apples in a bag and picking them up from the tree. |
| BART-base-VisCTG | A man puts a bag of apples on a tree. |
| Human reference | I picked an apple from the tree and put it in my bag. |

Table 6.11: Qualitative examples for test$_{CG}$. *BL* stands for baseline. *Concept set* refers to the input keywords and *Captions* refers to the captions (separated by <s>) used by the VisCTG model for that particular example to produce its final generation.

| **Model\Metrics** | ROUGE-2/L | | BLEU-3/4 | | METEOR | CIDEr | SPICE | BERTScore | Cov |
|---|---|---|---|---|---|---|---|---|---|
| Reported BART-large | 22.13 | 43.02 | 37.00 | 27.50 | 31.00 | 14.12 | 30.00 | - | 97.56 |
| Reported T5-base | 15.33 | 36.20 | 28.10 | 18.00 | 24.60 | 9.73 | 23.40 | - | 83.77 |
| Reported T5-Large | 21.98 | 44.41 | 40.80 | 30.60 | 31.00 | 15.84 | 31.80 | - | 97.04 |
| Our BART-base | 15.91 | 36.15 | 38.30 | 28.30 | 30.20 | 15.07 | 30.35 | 58.26 | 93.44 |
| Our BART-large | 17.27 | 37.32 | **39.95** | **30.20** | **31.15** | **15.72** | **31.20** | 58.58 | 95.03 |
| Our T5-base | **17.27** | **37.69** | **41.15** | **31.00** | **31.10** | **16.37** | **32.05** | 60.32 | **94.44** |
| Our T5-large | 17.90 | 38.31 | **43.80** | **33.60** | **32.70** | **17.02** | **33.45** | 61.39 | 96.26 |

Table 6.12: Performance of our re-implemented Commongen models on $\text{dev}_O$ compared to the original numbers reported in Lin et al. [99]. Note that for our models, results are averaged over two seeds, and that the original authors did not experiment with BART-base or report BERTScore. Bold indicates where we match or exceed the corresponding reported baseline metric.

| Method | Text |
|---|---|
| Concept set | {sunglass, wear, lady, sit} |
| Captions | a woman sitting on a bench with a cell phone <s> a woman sitting on a bench with a book |
| T5-base-BL | A lady sits in a sunglass. |
| T5-base-VisCTG | A lady wearing sunglasses sits on a bench. |
| Human reference | The lady wants to wear sunglasses, sit, relax, and enjoy her afternoon. |
| Concept set | {music, dance, room, listen} |
| Captions | a person is standing in a room with a bed <s> a woman is holding a laptop in a room |
| BART-large-BL | A listening music and dancing in a dark room |
| BART-large-VisCTG | A group of people are dancing and listening to music in a room. |
| Human reference | A boy danced around the room while listening to music. |
| Concept set | {pool, water, slide, slide} |
| Captions | a boat is parked in a water with a boat |
| T5-large-BL | A girl slides into a pool and slides into the water. |
| T5-large-VisCTG | A group of people slide down a slide into a pool of water. |
| Human reference | A boy slides down a bouncy slide into a pool of water. |
| Concept set | {rock, water, stand, body} |
| Captions | a bird sitting on a rock in a body of water |
| T5-large-BL | a body of water standing on rocks |
| T5-large-VisCTG | A man standing on a rock near a body of water. |
| Human reference | A bird standing on a large rock in a body of water. |
| Concept set | {card, deck, shuffle, hand} |
| Captions | a person holding a cell phone in their hand <s> a person holding a pair of scissors in their hand |
| BART-large-BL | a hand shakes a deck of cards |
| BART-large-VisCTG | A man shuffles a deck of cards with his hand. |
| Human reference | A man shuffles a deck of cards in his hands. |
| Concept set | {chase, ball, owner, dog, throw} |
| Captions | a dog is standing in the grass with a frisbee <s> a dog is playing with a frisbee in the grass |
| T5-base-BL | owner throws a ball to his dog during a chase. |
| T5-base-VisCTG | A dog is throwing a ball at its owner. |
| Human reference | The owner threw the ball for the dog to chase after. |
| Concept set | {body, water, bench, sit} |
| Captions | a bench sitting on a beach next to a body of water <s> a man is sitting on a bench with a cell phone <s> a bench sitting on a of a beach <s> a bench sitting in the middle of a lake <s> woman sitting on a bench with a bird in the background |
| BART-base-BL | A woman sitting on a bench with water in her body. |
| BART-base-VisCTG | A man sits on a bench near a body of water. |
| Human reference | The woman sat on the bench as she stared at the body of water. |
| Concept set | {bench, sit, talk, phone} |
| Captions | a man sitting on a bench with a cell phone <s> a woman sitting on a bench with a cell phone <s> a man sitting on a bench with a cell phone <s> a person sitting on a bench with a skateboard <s> a man sitting on a bench with a laptop |
| BART-base-BL | A man sitting on a bench talking to his phone. |
| BART-base-VisCTG | A man sitting on a bench talking on his cell phone. |
| Human reference | The woman sits on the bench to talk on her daughter on the phone. |

Table 6.13: Further qualitative examples for test$_{CG}$. *BL* stands for baseline. *Concept set* refers to the input keywords and *Captions* refers to the captions (separated by <s>) used by the VisCTG model for that particular example to produce its final generation.

# Chapter 7

# Viable Content Selection and Refex Generation Through Pragmatic Backoff For Chess Commentary Generation (ACL 2018)

> When one shows someone the king in chess and says: "This is the king", this does not tell him the use of this piece — unless he already knows the rules of the game up to this last point: the shape of the king. You could imagine his having learnt the rules of the game without ever having been shown an actual piece. The shape of the chessman corresponds here to the sound or shape of a word.
>
> Ludwig Wittgenstein, *Philosophical Investigations*

In this chapter, we introduce and examine the NLG setting of generating natural language descriptions of chess game moves. This setting requires the NLG model to learn to select content from a vast space of possible things to say given the current game state and move. Furthermore, the model also needs to learn to integrate selected content appropriately with preceding content.

However, the scale of training data available is insufficient to support acquiring both these abilities tabula rasa.

Our CG is to generate a short, interesting commentary after a given single move in an ongoing chess game, given the pre-move and post-move board states as input information. For studying this setting, we introduce a new large-scale chess commentary dataset and devise methods to viably generate commentary for individual moves in a chess game inspite of the underspecified CG and data availability that is insufficient to learn the game pragmatics tabula rasa. The introduced dataset consists of more than 298K chess move-commentary pairs across 11K chess games.

Consider the various repurcussions that a single move in a chess game engenders. Even a single move can change many inter-piece relationships and piece states in the game, including those between pieces that did not themselves change position during the move. (e.g., a black rook can threaten the white knight once a black knight blocking the horizontal path between them moved). The NLG model faces three key challenges:

1. What type of comment to make? One can describe the move and the game itself (*Move Description*), describe the quality of the move (*Move Quality*). We assume the desired comment type to be additionally given as part of the input, and augment our dataset input to address the same.
2. What to comment on out of the many updated states and relationships so that its interesting from the game's perspective? This is related to the *content selection* subtask that in turn falls in the *macroplanning* stage.
3. How to address and refer to the interesting pieces and their relationships in an interesting way from the game's perspective? This is related to the *referring expression generation* subtask from microplanning.

In order to address challenges 2 and 3, the NLG model encoder has to learn to encode the $>2^{8\times8}$ board states in a game-pragmatically sensitive way. Furthermore, it has to accomplish this *tabula rasa*, without any prior knowledge of the game's rules.

Acquiring a game-sensitive, pragmatic understanding of the input state is essential to solve both the macroplanning and microplanning challenges involved. Performing inadequately on either of these challenges leads to the trap of generating *common or dull language* [35, 177], that would not satisfy the communicative goal. We find that acquiring such an understanding tabula rasa is too challenging for a typical attentional LSTM-based encoder-decoder model instantiating E2EN2PP, leading to the *common response problem* as anticipated, and consequently, inferior performance to even template based baselines on both automatic and human metrics. Thus,

characteristic of the class of settings we study in this Part, the extent of training data available is insufficient to acquire the complex aspect required by the CG tabula rasa. Further, the CG itself in explicit terms is underspecified, and does not provide additional information about chess game pragmatics, whether through symbolic means e.g., a knowledge graph enlisting typical piece-piece configurations, or otherwise.

Hence, there arises a need to incorporate an External Knowledge Source to bridge this knowledge gap, which in this case takes the form of a game library capable of guiding featurization.

Based on our observations about ouput deficiencies, we devise an alternative model that includes an additional *Pragmatic Interpretation Layer* to discretely featurize the board states using a game library, essentially backing off to pragmatic game knowledge to viably declutter the input states, thereby simplifying the understanding and overcoming the microplanning and macroplanning issues observed. Consequently, the model is now able to outperform all baselines, including the template-based one, on both automatic and human metrics.

The devised Intervention that needs to be done in the E2EN2PP can be seen in Figure 7.1

Through a human study on predictions for a subset of the data that deals with direct move descriptions, we observe that outputs from our models are rated similar to ground truth commentary texts in terms of correctness and fluency.

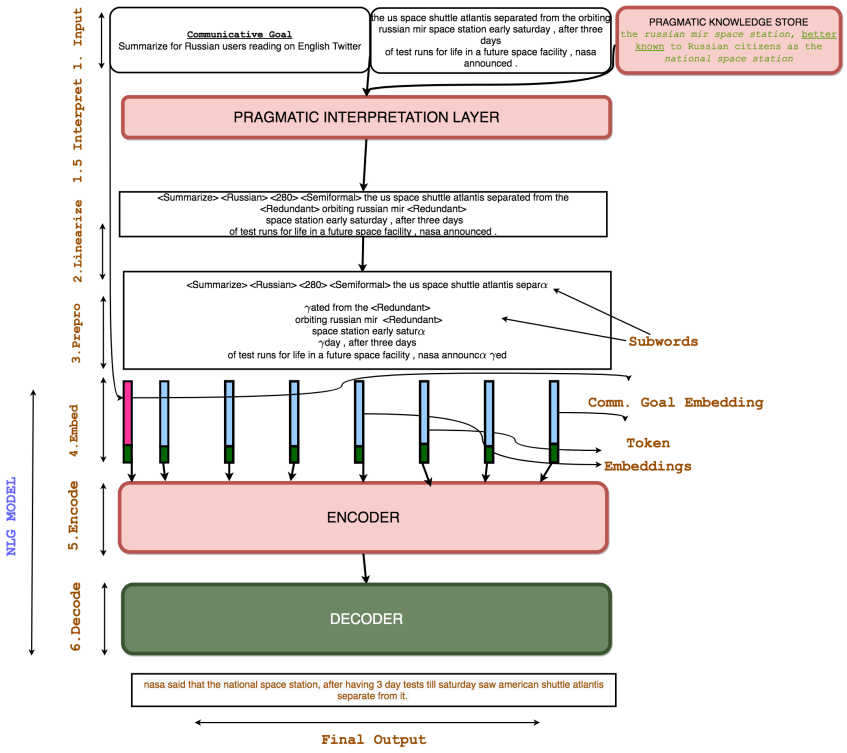


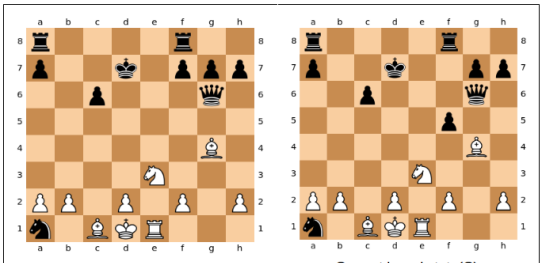

Figure 7.1: An illustration of how *End-to-End Neural NLG Pseudo-Pipeline* would work in action for an actual generation task and input example, after incorporating the Intervention in Chapter 7. Here, the task is to summarize the given input news article to within 280 characters. Note that this is a *Pseudo-Pipeline*, since the layers do not correspond to sub-tasks of NLG; moreover, they cannot be learnt or updated independently. The specific intervention shown here is the introduction of a *Pragmatic Interpretation Layer* that takes in the raw board states and featurizes them into a collection of discrete game-pertinent features.

## 7.1 Introduction

A variety of work in NLP has sought to produce fluent natural language descriptions conditioned on a contextual grounding. For example, several lines of work explore methods for describing images of scenes and videos [80], while others have conditioned on structured sources like Wikipedia infoboxes [93]. In most cases, progress has been driven by the availability of large training corpora that pair natural language with examples from the grounding [101]. One line of work has investigated methods for producing and interpreting language in the context of a game, a space that has rich pragmatic structure, but where training data has been hard to come by.

In this chapter, we introduce a new large-scale resource for learning to correlate natural language with individual moves in the game of chess. We collect a dataset of more than 298K

chess move/commentary pairs across ≈ 11K chess games from online chess forums. To the best of our knowledge, this is the first such dataset of this scale for a game commentary generation task. We provide an analysis of the dataset and highlight the large variety in commentary texts by categorizing them into six different aspects of the game that they respectively discuss.




**Template**: Black moves the pawn from f7 to f5

**Nearest Neighbor**: Attacks the bishop , but allows black to gain the pawn back .

**GAC(Our method)**: Black threatens white bishop with his pawn.

**Ground Truth**: Blocking the bishop out and eyeing for f4 and mate on c

Figure 7.2: Move commentary generated from our method (Game-aware neural commentary generation (GAC)) and some baseline methods for a sample move.

Automated game commentary generation can be a useful learning aid. Novices and experts alike can learn more about the game by hearing explanations of the motivations behind moves, or their quality. In fact, on sites for game aficionados, these commentaries are standard features, speaking to their interestingness and utility as complements to concrete descriptions of the game boards themselves.

Game commentary generation poses a number of interesting challenges for existing approaches to language generation. First, modeling human commentary is challenging because human commentators rely both on their prior knowledge of game rules as well as their knowledge of effective strategy when interpreting and referring to the game state. Secondly, there are multiple aspects of the game state that can be talked about for a given move — the commentator's choice depends on the pragmatic context of the game. For example, for the move shown in Figure 7.2, one can comment simply that the pawn was moved, or one may comment on how the check was blocked by that move. Both descriptions are true, but the latter is most salient given the player's goal. However, sometimes, none of the aspects may stand out as being most salient, and the most salient aspect may even change from commentator to commentator. Moreover, a human commentator may introduce variations in the way he or she chooses to talk about these aspects, in order to reduce monotony in the commentary. This makes the dataset a useful testbed for the

content selection and referring expression sub-skills of NLG.

There has been some, albeit very limited, prior work which has explored game commentary generation. [98, 152] have explored chess commentary generation, but for lack of large-scale training data their methods have been mainly based on rules defined manually. [79] have explored commentary generation for the game of Shogi, proposing a two-step process where salient terms are generated from the game state and then composed in a language model. In contrast, given the larger amount of training data available to us, our devised model uses a trainable neural architecture to predict commentaries given the game state. Our model conditions on semantic and pragmatic information about the current state and explicitly learns to compose, conjoin, and select these features in a recurrent decoder module. We perform an experimental evaluation comparing against baselines and variants of our model that ablate various aspects of our devised architecture. Outputs on the 'Move Description' subset of data from our final model were judged by humans to be as good as human written ground truth commentaries on measures of fluency and correctness.

## 7.2 Chess Commentary Dataset

In this section we introduce our new large-scale *Chess Commentary* dataset, share some statistics about the data, and discuss the variety in type of commentaries. The data is collected from the online chess discussion forum `gameknot.com`, that features multiple games self-annotated with move-by-move commentary.

The dataset consists of 298K aligned game move/commentary pairs. Some commentaries are written for a sequence of few moves while others correspond to a single move. For the purpose of initial analysis and modeling, we limit ourselves to only those data points where commentary text corresponds to a single move. Additionally, we split the multi-sentence commentary texts to create multiple data points with the same chess board and move inputs.

**What are commentaries about?** We observe that there is a large variety in the commentary texts. To analyze this variety, we consider labelling the commentary texts in the data with a predefined set of categories. The choice of these categories is made based on a manual inspection of a sub-sample of data. We consider the following set of commentary categories (Also shown in Table 7.2):

- **Direct move description (MoveDesc[2]):** Explicitly or implicitly describe the current move.
- **Quality of move (Quality[3]):** Describe the quality of the current move.

[2]MoveDesc & 'Move Description' used interchangeably
[3]Quality and 'Move Quality' used interchangeably

| Statistic | Value |
|---|---|
| Total Games | 11,578 |
| Total Moves | 298,008 |
| Average no. of recorded steps in a game | 25.73 |
| Frequent Word Types[1] | 39,424 |
| Rare Word Types | 167,321 |
| Word Tokens | 6,125,921 |
| Unigram Entropy | 6.88 |
| Average Comment Length (in #words) | 20.55 |
| Long Comments (#words $>$ 5) | 230745 (77%) |

Table 7.1: Dataset and Vocabulary Statistics

| Category | Example | % in data | Validation accuracy |
|---|---|---|---|
| Direct Move Description | An attack on the queen | 31.4% | 71% |
| Move Quality | A rook blunder. | 8.0% | 90% |
| Comparative | At this stage I figured I better move my knight. | 3.7% | 77.7% |
| Planning / Rationale | Trying to force a way to eliminate d5 and prevent Bb5. | 31.2% | 65% |
| Contextual Game Info | Somehow, the game I should have lost turned around in my favor . | 12.6% | 87% |
| General Comment | Protect Calvin , Hobbs | 29.9% | 78% |

Table 7.2: Commentary texts have a large variety making the problem of content selection an important challenge in our dataset. We classify the commentaries into 6 different categories using a classifier trained on some hand-labelled data, a fraction of which is kept for validation. % data refers to the percentage of commentary sentences in the tagged data belonging to the respective category.

- **Comparative:** Compare multiple possible moves.
- **Move Rationale or Planning (Planning):** Describe the rationale for the current move, in terms of the future gameplay, advantage over other potential moves etc.
- **Contextual game information:** Describe not the current move alone, but the overall game state, such as possibility of win/loss, overall aggression/defence, etc.
- **General information:** General idioms and advice about chess, information about players/-tournament, emotional remarks, retorts, etc.

The examples in Table 7.2 illustrate these classes. Note that the commentary texts are not necessarily limited to one tag, though that is true for most of the data. A total of 1K comments are annotated by two annotators. A SVM classifier [127] is trained for each comment class, considering the annotation as ground truth and using word unigrams as features. This classifier is then used to predict tags for the train, validation and test sets. For the "Comparative" category, we found that a classifier with manually defined rules such as presence of word "better" performs better than the classifier, perhaps due to the paucity of data, and thus we use this instead . As can be observed in Table 7.2, the classifiers used are able to generalize well on the held out dataset.

Note that we choose to focus on the first three categories given the large amount of variance in comments from the remaining ones.

## 7.3 Game Aware Neural Commentary Generations (GAC)

Our dataset $D$ consists of data points of the form $(S_i, M_i, G_i), i \in \{1, 2, .., |D|\}$, where $S_i$ is the commentary text for move $M_i$ and $G_i$ is the corresponding chess game (i.e., its game history/context). $S_i$ is a sequence of $m$ tokens $S_{i1}, S_{i2}, ..., S_{im}$. We want to model $P(S_i|M_i, G_i)$. For simplicity, we use only current board ($C_i$) and previous board ($R_i$) information from the game. $P(S_i|M_i, G_i) = P(S_i|M_i, C_i, R_i)$.

We model this using an end-to-end trainable neural model that models conjunctions of features using feature encoders. Our model employs a selection mechanism to select the salient features for a given chess move. Finally a LSTM recurrent neural network [66] is used to generate the commentary text based on selected features from encoder.

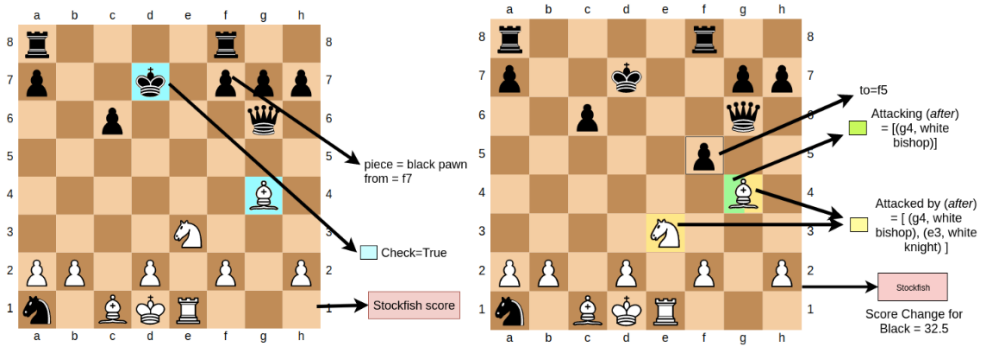



Figure 7.3: The figure shows some features extracted using the chess board states before (*left*) and after (*right*) a chess move. Our method uses various semantic and pragmatic features of the move, including the location and type of piece being moved, which opposing team pieces attack the piece being moved before as well as after the move, the change in score by *Stockfish* UCI engine, etc.

### 7.3.1 Incorporating Pragmatic Knowledge

Past work shows that acquiring pragmatic knowledge is critical for NLG systems [109, 144], particularly data-to-text NLG systems where the data is . Commentary texts cover a range of perspectives, including criticism or goodness of current move, possible alternate moves, quality of alternate moves, etc. To be able to make such comments, the model must learn about the quality of moves, as well as the set of valid moves for a given chess board state. We consider the following features to provide our model with necessary information to generate commentary texts (Figure 7.3):

**Move** features $f_{move}(M_i, C_i, R_i)$ encode the current move information such as which piece moved, the position of the moved piece before and after the move was made, the type and position of the captured piece (if any), whether the current move is castling or not, and whether there was a check or not.

**Threat** features $f_{threat}(M_i, C_i, R_i)$ encode information about pieces of opposite player attacking the moved piece before and after the move, and the pieces of opposite player being attacked by the piece being moved. To extract this information, we use the *python-chess*[4] library

**Score** features $f_{score}(M_i, C_i, R_i)$ capture the quality of move and general progress of the game. This is done using the game evaluation score before and after the move, and average rank of pawns of both the players. We use *Stockfish* evaluation engine to obtain the game evaluation scores. [5]

[4] https://pypi.org/project/python-chess/
[5] https://stockfishchess.org/about/

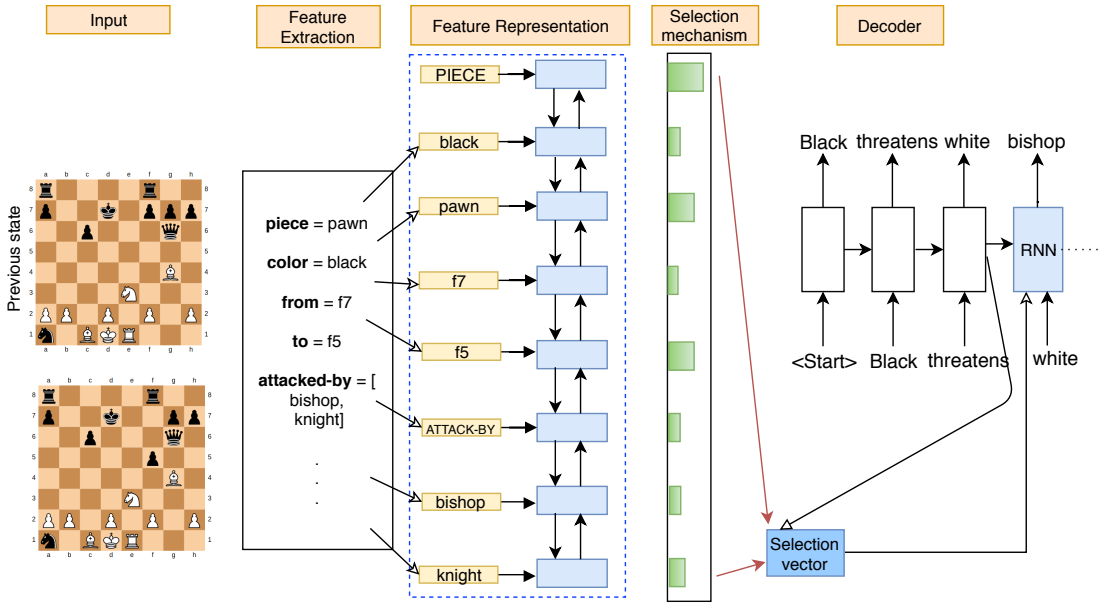



Figure 7.4: The figure shows a model overview. We first extract various semantic and pragmatic features from the previous and current chess board states. We represent features through embedding in a shared space. We observe that feeding in feature conjunctions helps a lot. We consider a selection mechanism for the model to choose salient attributes from the input at every decoder step.

### 7.3.2 Feature Representation

In our simplest conditioned language generation model **GAC-sparse**, we represent the above described features using sparse representations through binary-valued features $g_{sparse}(M_i, C_i, R_i) = \text{SparseRep}(f_{move}, f_{threat}, f_{score})$

For our full **GAC** model we consider representing features through embeddings. This has the advantage of allowing for a shared embedding space, which is pertinent for our problem since attribute values can be shared, e.g., the same piece type can occur as the moved piece as well as the captured piece. For categorical features, such as those indicating which piece was moved, we directly look up the embedding using corresponding token. For real valued features such as game scores, we first bin them and then use corresponding number for embedding lookup. Let $E$ represent the embedding matrix. Then $E[f^j_{move}]$ represents embeddings of $j^{th}$ move feature, or in general $E[f_{move}]$ represents the concatenated embeddings of all move features. Similarly, $E(f_{move}, f_{threat}, f_{score})$ represents concatenated embeddings of all the features.

### 7.3.3 Feature Conjunctions

We conjecture that explicitly modeling feature conjunctions might improve the performance. So we need an encoder that can handle input sets of features of variable length (features such as pieces attacking the moved piece can be of variable length). One way to handle this is by defining a canonical ordering of the features and consider a bidirectional LSTM encoder over the feature embeddings. As shown in Figure 7.4, this generates conjunctions of features.
$g^{enc} = \text{BiLSTM}^*(\{E(f_{move}, f_{threat}, f_{score}))\})$

Here $E()$ represents the embedding matrix as described earlier and $BiLSTM^*$ represents a sequential application of the $BiLSTM$ function. Thus, if there a total of $m$ pragmatics-aware feature keys and embedding dimension is $d$, $E(f_{move}, f_{threat}, f_{score})$ is matrix of $m * d$. If hidden size of BILSTM is of size $x$, then $g^{enc}$ is of dimensionality $m * x$. We observe that different orderings gave similar performance. We also experimented with running $k$ encoders, each on a different ordering of the pragmatics-aware features, and then letting the decoder access to each of the $k$ encodings. This did not yield any significant gain in performance.

The GAC model, unlike GAC-sparse, has some advantages as it uses a shared, continuous space to embed attribute values of different features, and can perform arbitrary feature conjunctions before passing a representation to the decoder, thereby sharing the burden of learning the necessary feature conjunctions. Our experiments confirm this intuition that GAC produces commentaries with higher BLEU as well as more diversity compared to GAC-sparse.

### 7.3.4 Decoder

We use a LSTM decoder to generate the sentence given the chess move and the features $g$. At every output step $t$, the LSTM decoder predicts a distribution over vocabulary words taking into account the current hidden state $h_t$, the input token $i_t$, and additional selection vector $c_t$. For GAC-sparse, the selection vector is simply an affine transformation of the features $g$. For GAC model selection vector is derived via a selection mechanism.

$$o_t, h_t^{dec} = LSTM(h_{t-1}^{dec}, [\text{concat}(E_{dec}(i_t), c_t)])$$
$$p_t = \text{softmax}(W_o[\text{concat}(o_t, c_t)] + b_s)$$

where $p_t$ represents th probability distribution over the vocabulary, $E_{dec}()$ represents the decoder word embedding matrix and elements of $W_o$ matrix are trainable parameters.

**Selection/Attention Mechanism:** As there are different salient attributes across the different

chess moves, we also equip the GAC model with a mechanism to select and identify these attributes. We first transform $h_t^{dec}$ by multiplying it with a trainable matrix $W_c$, and then take dot product of the result with each $g_i$.

$$a_t^{(i)} = dot(W_c * h_t^{dec}, g_i^{enc})$$
$$\alpha_t = \text{softmax}(a_t)$$
$$c_t = \sum_{i=1}^{i=|g|} \alpha_t^{(i)} g_i^{enc}$$

We use cross-entropy loss over the decoding outputs to train the model.

## 7.4 Experiments

We split each of the data subsets in a 70:10:20 ratio into train, validation and test. All our models are implemented in Pytorch version 0.3.1 [126]. We use the ADAM optimizer [84] with its default parameters and a mini-batch size of 32. Validation set perplexity is used for early-stopping. At test-time, we use greedy search to generate the model output. We observed that beam decoding does not lead to any significant improvement in terms of validation BLEU score.

We employ the BLEU [125] and BLEU-2 [176] scores to measure the performance of the models. Additionally, we consider a measure to quantify the diversity in the generated outputs. Note that we are aware of and acknowledge the many limitations of relying on BLEU as a metric for comment aptness, and use it primarily as an easy and tractable to compute metric during the model development. For final model comparison, in §7.4.5, we also request human annotations for multiple aspects of comment validity, which are a much more suitable evaluation approach than BLEU (though we further enlist and acknowledge this setup's own limitations in §7.6.1 and §7.6.2).

Finally, we also conduct a human evaluation study. In the remainder of this section, we discuss baselines along with various experiments and results.

### 7.4.1 Baselines

In this subsection we discuss the various baseline methods.

**Manually-defined template (TEMP)** We devise manually defined templates [141] for 'Move Description' and 'Move Quality' categories. Note that template-based outputs tend to be repetitive

as they lack diversity, drawing from a small, fixed vocabulary and using a largely static sentence structure. We define templates for a fixed set of cases which cover our data. (For exact template specifications, refer to Appendix B)

**Nearest Neighbor (NN):** We observe that the same move on similar board states often leads to similar commentary texts. To construct a simple baseline, we find the most similar move $N_{MCR}$ from among training data points for a given previous ($R$) and current ($C$) board states and move $M$. The commentary text corresponding to $N_{MCR}$ is selected as the output. Thus, we need to consider a scoring function to find the closest matching data point in training set. We use the *Move*, *Threat* and *Score* features to compute similarity to do so. By using a sparse representation, we consider total of 148 *Move* features, 18 *Threat* features, and 19 *Score* features. We use sklearn's [128] NearestNeighbor module to find the closest matching game move.

**Raw Board Information Only (RAW):** The RAW baseline ablates to assess the importance of our pragmatic feature functions. This architecture is similar to GAC, except that instead of our custom features $A(f(R_i, C_i))$, the encoder encodes raw board information of current and previous board states.
$A_{RAW}(R_i, C_i) = [Lin(R_i), Lin(C_i)]$
$Lin()$ for a board denotes its representation in a row-linear fashion. Each element of $Lin()$ is a piece name (e.g., *pawn*) denoting the piece at that square with special symbols for empty squares.

### 7.4.2 Comment Category Models

As shown earlier, we categorize comments into six different categories. Among these, in this chapter we consider only the first three as the amount of variance in the last three categories indicates that it would be extremely difficult for a model to learn to reproduce them accurately. The number of data points, as tagged by the trained classifiers, in the subsets 'Move Description', 'Move Quality' and 'Comparative' are 28,228, 793 and 5397 respectively. We consider separate commentary generation models for each of the three categories. Each model is tuned separately on the corresponding validation sets. Table 7.3 shows the BLEU and BLEU-2 scores for the devised model under different subsets of features. Overall BLEU scores are low, likely due to the inherent variance in NLG tasks such as dialog response generation and data-to-text description (of which our task is an example) generation tasks, where even adequate outputs sometimes do not match references due to many possible outputs being adequate for the same input [121]. A

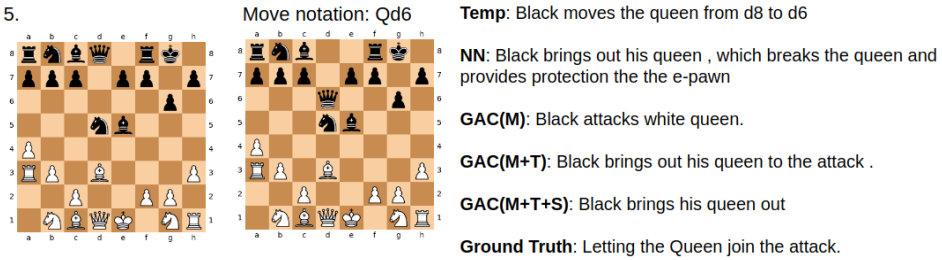



Figure 7.5: Outputs from various models on a test example from the **MoveDesc** subset.

precursory examination of the outputs for data points selected randomly from the test set indicated that they were reasonable. Figure 7.5 illustrates commentaries generated by our models through an example (a larger list of qualitative examples can be found in Appendix C).

**Which features are useful?** In general, adding *Threat* features improves the performance, though the same is not always true for *Score* features. One reason for the importance of *Threat* features might be that the change in value of a *Threat* feature often includes comment-worthy events such as the king coming under check, or going out of check. Learning to detect the activation and de-activation of a situation where one piece threatens another is challenging in the absence of *Threat* features, since the piece being threatened and the piece which is threatening may even both not be identical to the piece which was immediately moved. Making such inferences would require the model to not just memorize all the piece positions but also, after each move, to implicitly compute whether a threat is activated by iterating over all opposing piece pairs.

The changes in *Score* features may often correspond to subtle changes later in the game tree, and are perhaps unlikely to immediately trigger the use of specific phrases in resultant commentary. Furthermore, for lot of cases where they do trigger use of specific phrases, this might be due to activation or de-activation of threats, or some specific specialities of the move (e.g., Castling) which is already explicitly captured by the *Move* and *Threat* features. This might explain the observation that addition of *Score* features does not always improve performance.

*Qual* has higher BLEU scores than the other datasets due to smaller vocabulary and lesser variation in commentary. As can be observed in Table 7.4, *Threat* features are useful for both 'Move Quality' and 'Move Description' subsets of data. Adding *Score* features helps for 'Move Quality' subset. This intuitively makes sense since *Score* features directly encode proxies for move quality as per a chess evaluation engine.

| Dataset | Features | BLEU | BLEU-2 | Diversity |
|---|---|---|---|---|
| MoveDesc | TEMP | 0.72 | 20.77 | 4.43 |
| | NN (M+T+S) | 1.28 | 21.07 | **7.85** |
| | RAW | 1.13 | 13.74 | 2.37 |
| | GAC-sparse | 1.76 | 21.49 | 4.29 |
| | GAC (M+T) | **1.85** | **23.35** | **4.72** |
| Quality | TEMP | 16.17 | 47.29 | 1.16 |
| | NN (M+T) | 5.98 | 42.97 | 4.52 |
| | RAW | 16.92 | 47.72 | 1.07 |
| | GAC-sparse | 14.98 | 51.46 | 2.63 |
| | GAC(M+T+S) | 16.94 | 47.65 | 1.01 |
| Comparative | NN (M) | 1.28 | 24.49 | 6.97 |
| | RAW | 2.80 | 23.26 | 3.03 |
| | GAC-sparse | **3.58** | 25.28 | 2.18 |
| | GAC(M+T) | 3.51 | 29.48 | 3.64 |

Table 7.3: Performance of baselines and our model with different subsets of features as per various quantitative measures.
( **S** = Score, **M**= Move, **T** = Threat features; ) On all data subsets, our model outperforms the **TEMP** and **NN** baselines. Among devised models, GAC performs better than GAC-sparse & RAW in general. For NN, GAC-sparse and GAC methods, we experiment with multiple feature combinations and report only the best as per BLEU scores.

### 7.4.3 A Single Model For All Categories

In this experiment, we merge the training and validation data of the first three categories and tune a single model for this merged data. We then compare its performance on all test sentences in our data. **COMB** denotes using the best GAC model for a test example based on its original class (e.g., *Desc*) and computing the BLEU of the sentences so generated with the ground truth. **GAC-all** represents the GAC model learnt on the merged training data.

As can be seen from Table 7.5, this does not lead to any performance improvements. We investigate this issue further by analyzing whether the board states are predictive of the type of category or not. To achieve this, we construct a multi-class classifier using all the *Move*, *Threat* and *Score* features to predict the three categories under consideration. However, we observe accuracy of around 33.4%, which is very close to the performance of a random prediction model. This partially explains why a single model did not fare better even though it had the opportunity to learn from a larger dataset.

**Category-aware model (CAT)** We observed above that with the considered features, it is not possible to predict the type of comment to be made, and the GAC-all model results are better than

| Dataset | Features | BLEU | BLEU-2 | Diversity |
|---|---|---|---|---|
| MoveDesc | GAC (M) | 1.41 | 19.06 | 4.32 |
| | GAC (M+T) | **1.85** | **23.35** | **4.72** |
| | GAC (M+T+S) | 1.64 | 22.82 | 4.29 |
| Quality | GAC (M) | 13.05 | 48.37 | 1.61 |
| | GAC (M+T) | 14.22 | 49.57 | **1.54** |
| | GAC(M+T+S) | **14.44** | **51.79** | 1.48 |
| Comparative | GAC(M) | 3.10 | 19.84 | 2.88 |
| | GAC(M+T) | **3.51** | **29.48** | **3.64** |
| | GAC(M+T+S) | 1.15 | 25.44 | 3.14 |

Table 7.4: Performance of the GAC model with different feature sets. ( **S** = Score, **M**= Move, **T** = Threat features; ) Different subset of features work best for different subsets. For instance, *Score* features seem to help only in the Quality category. Note that the results for Quality are from 5-fold cross-validation, since the number of datapoints in the category is much lesser than the other two.

| Dataset | Features | BLEU | BLEU-2 | Diversity |
|---|---|---|---|---|
| All | COMB (M) | 2.07 | 20.13 | 4.50 |
| | COMB (M+T) | **2.43** | **25.37** | 4.88 |
| | COMB (M+T+S) | 1.83 | 28.86 | 4.33 |
| All | GAC-all(M) | 1.69 | 20.66 | 4.67 |
| | GAC-all(M+T) | 1.94 | 24.11 | 5.16 |
| | GAC-all (M+T+S) | 2.02 | 24.70 | 4.97 |
| All | CAT (M) | 1.90 | 19.96 | 3.82 |

Table 7.5: The **COMB** approaches show the combined performance of separately trained models on the respective test subsets.

COMB results. Hence, we extend the GAC-all model to explicitly provide with the information about the comment category. We achieve this by adding a one-hot representation of the category of the comment to the input of the RNN decoder at every time step. As can be seen in the Table 7.5, CAT(M) performs better than GAC-all(M) in terms of BLEU-4, while performing slightly worse on BLEU-2. This demonstrates that explicitly providing information about the comment category can help the model.

### 7.4.4 Diversity In Generated Commentaries

Humans use some variety in the choice of words and sentence structure. As such, outputs from rule based templates, which demonstrate low variety, may seem repetitive and boring. To capture this quantitatively, and to demonstrate the variety in texts from our method, we calculate the entropy [157] of the distribution of unigrams, bigrams and trigrams of words in the predicted

outputs, and report the geometric mean of these values. Using only a small set of words in similar counts will lead to lower entropy and is undesirable. As can be observed from Table 7.3, template baseline performs worse on the said measure compared to our methods for the 'MoveDesc' subset of the data.

### 7.4.5 Human Evaluation Study

As discussed in the qualitative examples above, we often found the outputs to be good, though BLEU scores are low. BLEU is known to correlate poorly [121, 142, 182] with human relevance scores for NLG tasks. Hence, we conduct a human evaluation study for the best 2 neural (GAC,GAC-sparse) and best 2 non-neural methods (TEMP,NN).

**Setup:** Specifically, annotators are shown a chess move through previous board and resulting board snapshots, along with information on which piece moved (a snapshot of a HIT[6] is provided in the Appendix D). With this context, they were shown text commentary based on this move and were asked to judge the commentary via three questions, shortened versions of which can be seen in the first column of Table 7.6.

We randomly select 100 data points from the test split of 'Move Description' category and collect the predictions from each of the methods under consideration. We hired two Anglophone (Lifetime HIT acceptance $\% > 80$) annotators for every human-evaluated test example. We additionally assess chess proficiency of the annotators using proficiency-evaluating questions (not to be confused with the questions we ask annotators during actual evaluation and annotation) from the chess-QA dataset by [24]. Within each HIT, we ask two randomly selected proficiency questions from the chess-QA dataset. Finally we consider only those HITs wherein the annotator was able to answer the proficiency questions correctly.

**Results:** We conducted a human evaluation study for the *MoveDesc* subset of the data. As can be observed from Table 7.6, outputs from our method attain slightly more favorable scores compared to the ground truth commentaries. This shows that the predicted outputs from our model are not worse than ground truth on the said measures. This is in spite of the fact that the BLEU-4 score for the predicted outputs is only $\sim 2$ w.r.t. the ground truth outputs. One reason for slightly lower performance of the ground truth outputs on the said measures is that some of the human written commentaries are either very ungrammatical or too concise. A more surprising observation is that

[6]Human Intelligence Task

| Question | GT | GAC (M) | GAC (MT) | GAC (MTS) | GAC -sparse | TEMP | NN |
|---|---|---|---|---|---|---|---|
| Qn: Is commentary correct for the given move? (%Yes) | 70.4 | 42.3 | 64.8 | 67.6 | 56.3 | 91.5 | 52.1 |
| Qn: Can the move be inferred from the commentary? (%Yes) | 45.1 | 25.3 | 42.3 | 36.7 | 40.8 | 92.9 | 42.3 |
| Fluency (scale of (least)1 - 5(most) ) Mean (Std. dev.) | 4.03 (1.31) | 4.15 (1.20) | 4.44 (1.02) | 4.54 (0.89) | 4.15 (1.26) | 4.69 (0.64) | 3.72 (1.36) |

Table 7.6: Human study results. Outputs from GAC are in general better than ground truth, NN and GAC-sparse. TEMP outperforms other methods, though as shown earlier, outputs from TEMP lack diversity.

around 30% of human written ground truth outputs were also marked as not valid for given board move. On inspection, it seems that commentary often contains extraneous game information beyond that of move alone, which indicates that an ideal comparison should be over commentary for an entire game, although this is beyond the scope of the current work.

The inter-annotator agreement for our experiments (Cohen's $\kappa$ [26]) is 0.45 for Q1 and 0.32 for Q2. We notice some variation in $\kappa$ coefficients across different systems. While TEMP and GAC responses had a 0.5-0.7 coefficient range, the responses for CLM had a much lower coefficient. In our setup, each HIT consists of 7 comments, one from each system. For Q3 (fluency), which is on an ordinal scale, we measure rank-order consistency between the responses of the two annotators of a HIT. Mean Kendall $\tau$ [82] across all HITs was found to be 0.39.

To measure statistical significance of results, we perform bootstrap tests on 1000 subsets of size 50 with a significance threshold of $p = 0.05$ for each pair of systems. For Q1, we observe that GAC(M), GAC(M+T) and GAC(M+T+S) methods are significantly better than baselines NN and GAC-sparse. We find that neither of GAC(M+T) and GT significantly outperform each other on Q1 as well as Q2. But we do find that GAC(M+T) does better than GAC(M) on both Q1 and Q2. For fluency scores, we find that GAC(M+T) is more fluent than GT, NN , GAC-sparse, GAC(M). Neither of GAC(M) and GAC(M+T+S) is significantly more fluent than the other.

## 7.5 Related Work

Data-to-text NLG research has a long and rich history, with systems ranging from completely rule-based [25] to learning-based ones [21, 143, 145], which have had both practical successes [145] and failures [143]. Recently, there have been numerous works that propose text generation given input information such as structured records, biographies [93], recipes [83, 186], etc. A key difference between generation given a game state compared to these inputs is that the game state is an evolving description at a point in a process, as opposed to recipes (that are independent

of each other), records (which are static) and biographies (which are one per person, and again independent). Moreover, our devised method effectively uses various types of semantic and pragmatic information about the game state.

In this chapter, we have introduced a new large-scale data for game commentary generation. The commentaries cover a variety of aspects like move description, quality of move, and alternative moves. This leads to a content selection challenge, similar to that noted in [182]. Unlike [182], our focus is on generating commentary for individual moves in a game, as opposed to game summaries from aggregate statistics as in their task.

One of the first NLG datasets was the SUMTIME-METEO [145] corpus with $\approx 500$ record-text pairs for technical weather forecast generation. Liang et al. [97] worked on common weather forecast generation using the WEATHERGOV dataset, that has $\approx$ 10K record-text pairs. A criticism of WEATHERGOV is that weather records themselves may have used templates and rules with optional human post-editing. There have been prior works on generating commentary for ROBOCUP matches [21, 112]. The ROBOCUP dataset, however, is collected from 4 games and contains about 1K events in total. Our dataset is two orders of magnitude larger than the ROBOCUP dataset, and we hope that it provides a promising setting for future NLG research.

## 7.6 Conclusions

In this chapter, we first introduce and study the Knowledge Deficient NLG setting of chess commentary generation. We also curate a dataset for this setting. We then motivate and devise Interventions to the E2EN2PP and resultant methods to perform generation on this dataset. At the outset, we find that our initial baseline model, that is a typical attentional LSTM-based encoder-decoder framework instantiating E2EN2PP, is found to lapse into merely generating input-dependent common responses, underperforming even template-based baselines. The model responses rarely choose the pertinent and interesting pieces as well as inter-piece relationships to talk about given the current move. They even fail to simply describe the piece that moved and its initial and final locations, as evinced by their underperforming even the template-based baseline which follows that simple strategy. This indicates their deficient performance on the content selection subskill. Furthermore, even when the responses choose the adequate piece (s) movements, and inter-piece relationships to describe, they seldom employs rich expressions such as *joining the attack*, *develops his position*, *putting in check* etc., indicating the model's deficiency at referring expression generation.

We posited that the aforementioned deficiencies were due to the model's inability to understand

the game states in the larger pragmatic context of the game. This motivated the need for an External Knowledge Source that could enhance the pragmatic context available.

Thus, we devised a method that directly provides knowledge about game pragmatics to its decoder via backing off to a game-library based discrete pragmatics-aware featurization introduced as an *Intervention* in the form of a *Pragmatic Interpretation Layer* (see §7.3.1 for more details on the features). This method overcomes aforementioned deficiencies, resulting in a viable commentary generator outperforming all the baselines, including the template-based one. The devised Intervention to the E2EN2PP, that causes a departure from the end-to-end nature, is illustrated in Figure 7.1.

Our devised method effectively utilizes information related to the rules and pragmatics of the game. A human evaluation study judges outputs from the devised methods to be as good as human written commentary texts.

Subsequent work [160] has proposed reinforcement learning based game-playing agents that learn to play board games from scratch, learning end-to-end from both recorded games and self-play. An interesting point to explore is whether such pragmatically trained game state representations can be leveraged for the task of game commentary generation.

### 7.6.1 Concerns/Deficiencies About the Overall Setting and its Evaluation

In this section, we outline and acknowledge limitations posed both by the way our overall CG, the setting and its experimental setup are framed as well as the accompanying human evaluation. These limitations restrict the extent to which the pragmatic relevance of the generated comments is assessed and conforms to its full, real-world form.

Predicting which type of comment to make for a given game situation/move is itself an important pragmatic consideration a real-world chess commentary system would need to address, and be assessed on. However, by including the commentary type as a explicitly specified control/input component, we make our setting and its experimental setup less capable of being assessed on this front.

Nonetheless, one potential workaround to still assess this aspect during human evaluation would have been to:

1. Present annotators with all three types of comments for the same input situation
2. Check how closely their preferences over the comment types as ranked along various aspects correlated with the model's own "confidences" about the comments it generated for each of the comment type (with confidence being evaluated by some suitable proxy notion, such as, e.g.,

the model probability of generated comment conditional on the input and respective comment type)

Our evaluation setup at the stage of initial writing and publication did not explore this workaround, and we acknowledge this as a limitation to be addressed.

### 7.6.2 Limitations w.r.t the Correctness/Aptness Distinction

Our human evaluation setup (see §7.4.5 and Table 7.6 for more ) required annotators to assess model generated comments along two questions besides mere fluency.

1. Is commentary correct for the given move?
2. Can the move be inferred from the commentary?

These two questions are largely sufficient to assess both correctness and specificity of the comment given the input situation. However, they are still insufficient at covering the notion of pragmatic relevance, which goes beyond mere correctness and specificity.

### 7.6.3 Broader Takeaways

Whenever there is a sharp disparity between the granularity at which the input (or the input portion of the communicative goal) states information and the granularity at which the output is expected to operate, one expects a gap between the understanding module and the generation module's representations, and the generation module has the added burden of learning to map the more disparate understanding representation to its own representation.

If filling this gap requires extensive acquisition of commonsense or other forms of knowledge (such as knowledge which can only be acquired through gameplay as in our case), it is possible that the above mappings may be deficient. In such a case, to perform NLG subtasks such as content selection viably (which may otherwise be obstructed), it becomes necessary to devise some way of converting the input to the right granularity by incorporating this knowledge either completely or through some heuristics, thus bridging the granularity gap.

Consider the example of generating game summary commentary for sports such as soccer and American football, given only the various summary scores and highlights tables generated in aggregate at the end of the game. These tables may contain a wide, heterogenous range of information, with only limited lexical interpretation offered by the row and column names, which themselves are domain-specific terms such as *Home/Away*, *Possession*, including even acronyms such as *GA* (Goals Attempted), etc. Different parts of this information may become pertinent

for different games. For example, *Away* games are generally harder for visiting teams, and even a closely fought draw or loss may entail not entirely negative commentary towards the visiting team e.g., *In a hard fought game, . . .* , *In a close encounter in harsh conditions, . . .* . A NLG model which attempts to learn the game summary commentary task may have a hard time learning to select, tabula rasa, the right content by decluttering and combining the appropriate row and column tuples.

However, if one were to instead introduce a Interpretation layer, like in this chapter, which first does either or both of the below pre-processing steps would greatly ease the model's ability to learn to content select and refer to game information appropriately.

1. Create a large number of game-pertinent, discrete 0-1 or multi-valued features, which it extracts from the game tables. e.g., a 0-1 feature can be Away games lost with a margin of only 2 goals
2. Perform preliminary lexicalization which uses simple rules to convert the table cells into simple lexical statements e.g., *Team X had a formation of 4 defenders, 4 midfielders and 2 strikers*. These statements can be included as additional parts of the input.

# Appendix A: Additional Data Examples

# Appendix B: Additional details for methods

## Templates

- *Move Description:* For the Move Description category, we consider following templates:
    1. **Capture** moves : [PLAYERMOVED] captures the [CAPTUREDPIECE] at [FINALSQUARE] using the [PIECEMOVED] at [INITIALSQUARE].
    2. **Non-Capture** moves: [PLAYERMOVED] moves the [PIECEMOVED] from [INITIALSQUARE] to [FINALSQUARE].
    3. **Castling** moves: [PLAYERMOVED] does a castling.

    For moves which lead to a CHECK in the resultant board state, an additional *putting the king in check* is added to the template. [PLAYERMOVED] (Black/White), [INITIALSQUARE], [FINALSQUARE], [PIECEMOVED] are filled in based on the move description on the input side.

| Text | Categories |
|---|---|
| Unpins and defends the knight , but it<br>does n't matter , as the time is ripe . | Desc |
| He gets fed up and exchanges Queen for Rook . | Desc |
| Rxc3 , I just retake with my queen , whilst if he<br>attempts defense with the bishop , then after 17.Bd2 ,<br>Ne4 , 18.Rxc3 , Nxg3 , 19.Rxc6 , Nxh1 , I 've won a rook outright . | Desc,Rationale |
| Preparing to castle , and threatening<br>now white 's e pawn for real. | Desc |
| Simply getting my rook off that dangerous diagonal<br>, and protecting the b pawn . | Desc |
| I throw in a check | Desc |
| Threatening mate with Qxh2 | Desc,Quality |
| A punch drunk move ! | Quality |
| This is not the best move. | Quality |
| The most logical move. | Quality |
| This move is dubious. | Quality |
| The check gains time to support the advance of the a-paw | Desc,Quality |
| maybe Ke1 was better | Rationale |
| I did n't want to retreat the N and I rejected 11 . | Rationale |
| I wish to both defend the pawn , and threaten indirectly the<br>black queen , gaining a tempo | Rationale |
| it would suite me better if my opponent made a queenside castling , since<br>then my advanced pawn on the d-file would assist in a future attack on the king 's position . | Comparative |
| but better would be nd2 to get the knight in the game , the queen rook , too . | Comparitive |
| i think it would have been better to play nxe5 and maintain a material advantage . | Comparitive |
| although not as effective as the bishop move , even 10.0-0-0 is better than the text ,<br>though 10 ... bg4 would have been very nasty . | Comparitive |
| fianchettoing , so that when black does complete his development , his b will be on a better diagnol . | Comparitive |
| He doesn't notice that his Knight is hanging ... | GameInfo |
| Now of course my forces are anchored around the pawns on e3 and h5 , and the black rook<br>loses his hope of penetrating the white position on the e-file | GameInfo |
| Well, now the game will get interesting soon | GeneralInfo |
| He tries his trick , which of course is noticed | GeneralInfo |
| This is often what I will do , when I 'm playing white. | GeneralInfo |

Table 7.7: Some commentary texts from each of the six categories. The **Categories** column lists those into which the example falls. As pointed out earlier, the category labels are not exclusive i.e., a text can belong to multiple categories, though texts with more than one category are few in our dataset. ('Desc' is shor for 'Move Description')

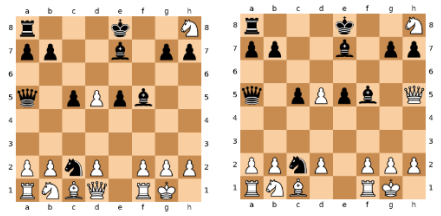
- *Move Quality:* Based on the move score ( as calculated by the chess engine *Stockfish*) $> \theta$ or $< \theta$, one of the following two is generated:

    1. A good move.
    2. A bad move. The threshold $\theta$ is found by tuning it on the validation set to maximize BLEU. We start from $\theta = 0$.

# Appendix C: Qualitative examples

Some qualitative examples are illustrated next:

2.




**Temp**: White moves the queen from d1 to h5 , putting the king in check

**NN**: White 's Queen strikes with 'Check ! '

**GAC(M)**: White throws in the check.

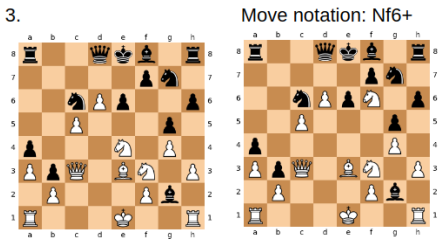
**GAC(M+T)**: White takes the bishop with his queen.

**GAC(M+T+S)**:White checks him with his queen..

**Ground Truth**: I put the king in check.

Figure 7.6: Example output 1: Move description subset of data.

3.




**Temp**: White moves the knight from e4 to f6 , putting the king in check

**NN**: { 13 : +3.25 } 23

**GAC(M)**: White sees the threat and threatens the knight.

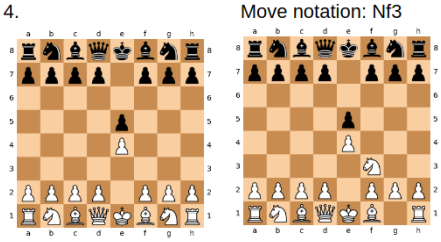
**GAC(M+T)**: White throws in a check.

**GAC(M+T+S)**: White throws in a check.

**Ground Truth**:The obvious move , forcing off the black queen and maintaining the menacing pawn presence on c5 and d6 , white 's attack looks dangerously close to being winning .

Figure 7.7: Example output 2: Move description subset of data.

4.




**Temp**: White moves the knight from g1 to f3

**NN**: I bring out my knight to f3 as usual.

**GAC(M)**: White develops his knight.

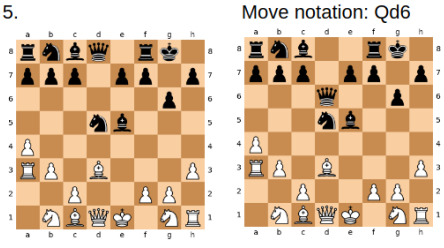
**GAC(M+T)**: White brings out his knight to attack the pawn.

**GAC(M+T+S)**: White develops his knight.

**Ground Truth**:Maintaining the tempo , and making the opponent respond to my move

Figure 7.8: Example output 3: Move description subset of data.

5.




**Temp**: Black moves the queen from d8 to d6

**NN**: Black brings out his queen , which breaks the queen and provides protection the the e-pawn

**GAC(M)**: Black attacks white queen.

**GAC(M+T)**: Black brings out his queen to the attack .

**GAC(M+T+S)**: Black brings his queen out

**Ground Truth**: Letting the Queen join the attack.

Figure 7.9: Example output 4: Move description subset of data.

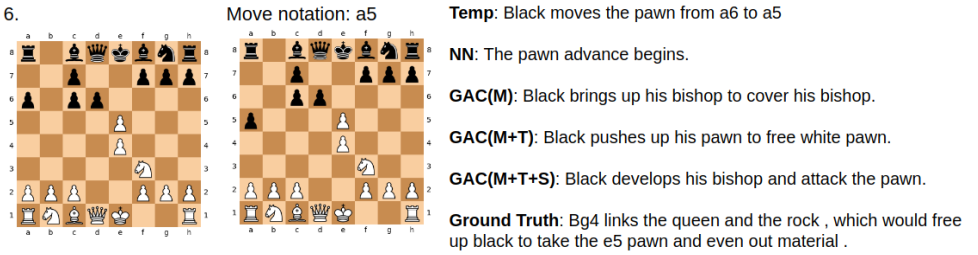



Figure 7.10: Example output 5: Move description subset of data.

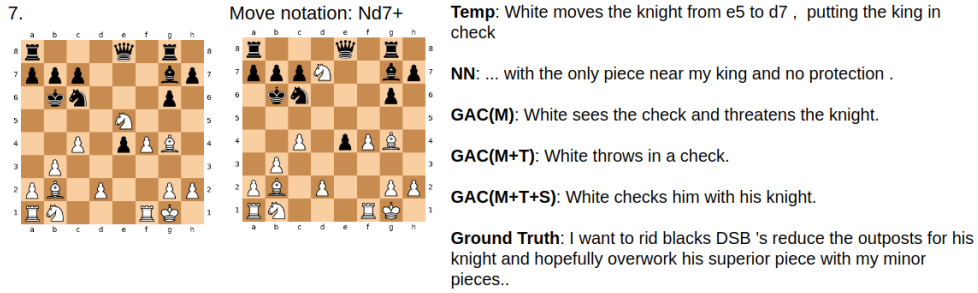



Figure 7.11: Example output 6: Move description subset of data.

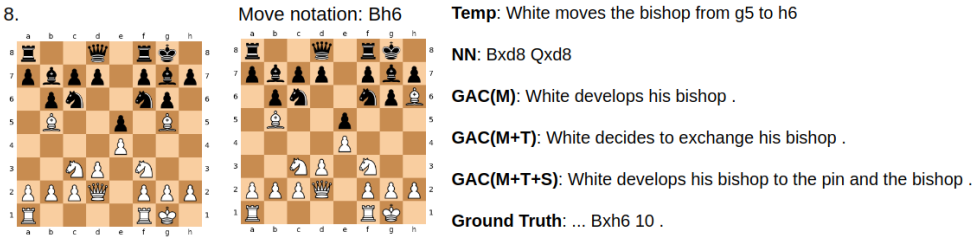



Figure 7.12: Example output 7: Move description subset of data.

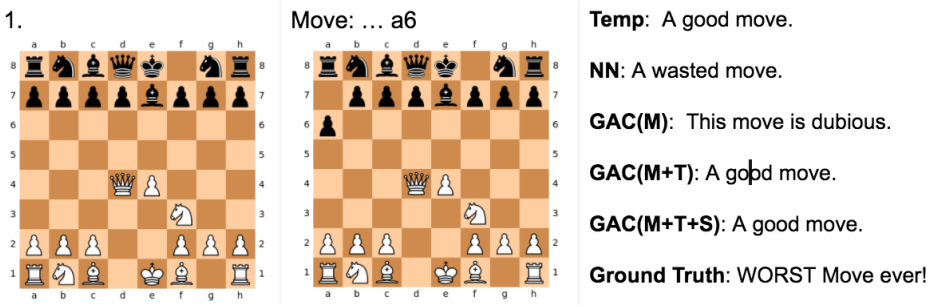



Figure 7.13: Example output 1: Move quality subset of data.

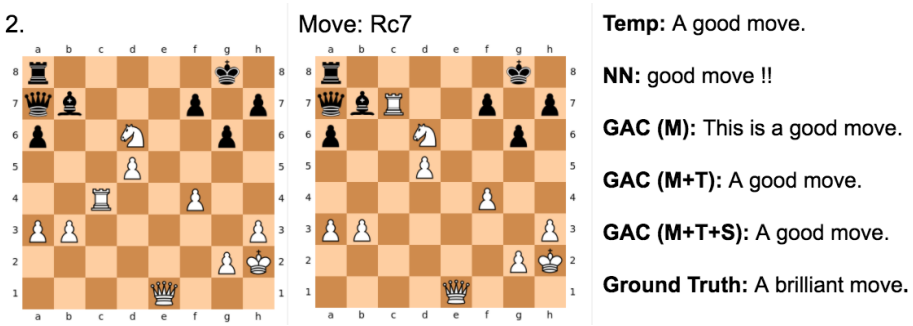



Figure 7.14: Example output 2: Move quality subset of data.

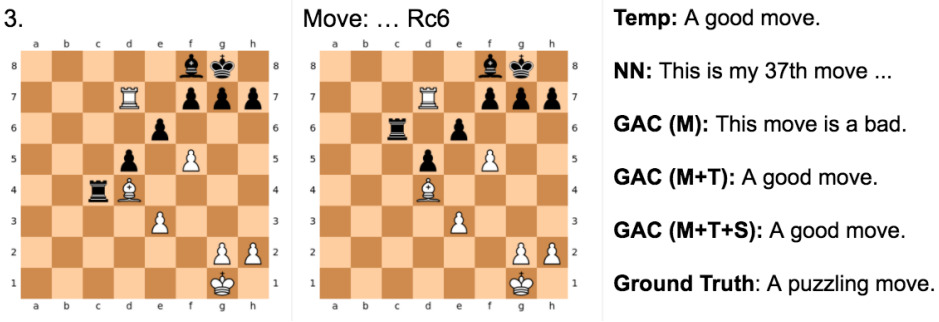



Figure 7.15: Example output 3: Move quality subset of data.

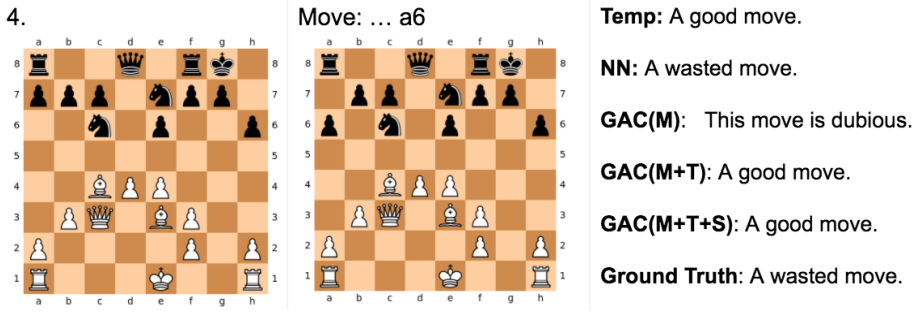



Figure 7.16: Example output 4: Move quality subset of data.

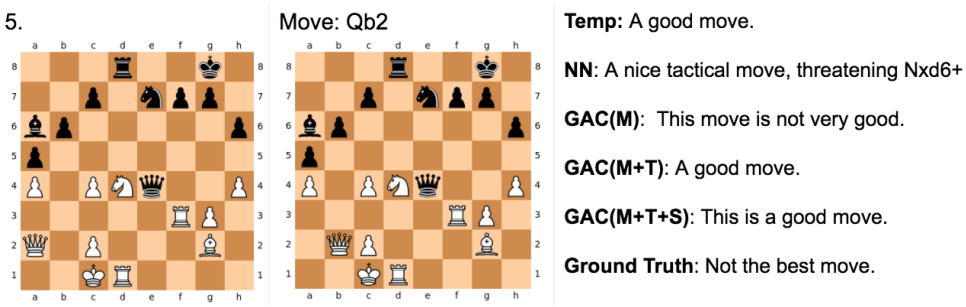



Figure 7.17: Example output 5: Move quality subset of data.

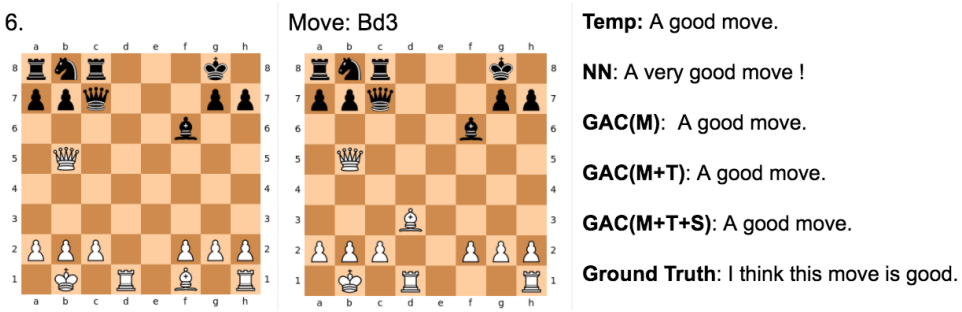



Figure 7.18: Example output 6: Move quality subset of data.

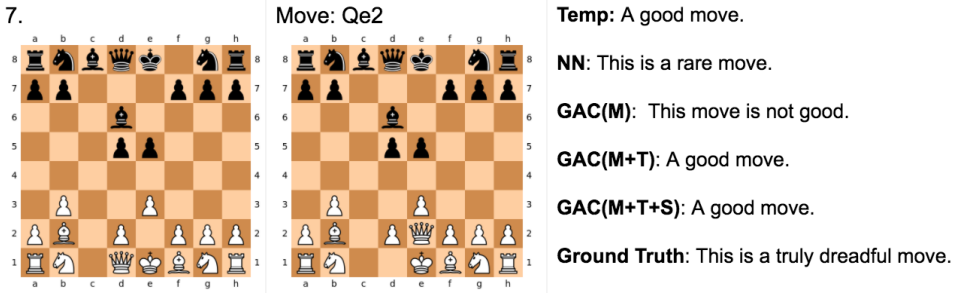



Figure 7.19: Example output 7: Move quality subset of data.

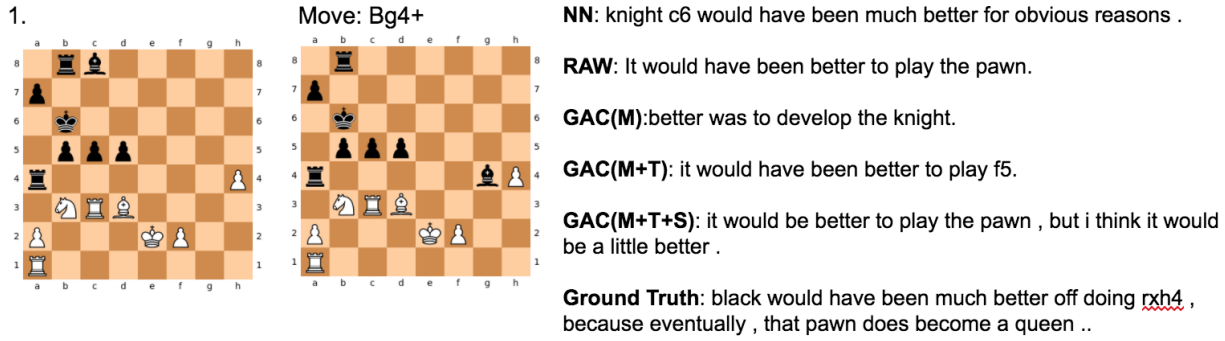



Figure 7.20: Example output 1: Comparative subset of data.

2.

Move: Re8

**NN**: what better way to defend than to attack.

**RAW**: better would have been to retreat the knight.

**GAC(M)**: that would have been better.

**GAC(M+T)**: it would have been better to play the rook.

**GAC(M+T+S)**: it would be better to play the knight to move to the king , but i think it would be better to play the knight to move to the king side.

**Ground Truth**: as better would 've been to retreat her ra5.

Figure 7.21: Example output 2: Comparative subset of data.

3.

Move: Rc8

**NN**: i think c3 was better.

**RAW**: better would have been to retreat the bishop to retreat.

**GAC(M)**:that would have been better.

**GAC(M+T)**: it would have been better to do the bishop to the king side.

**GAC(M+T+S)**: it would be better to play the knight to move to the knight , but i wanted to play the knight to move to the king side of the board.

**Ground Truth**: understandable , but 5 bc4 might be better.

Figure 7.22: Example output 3: Comparative subset of data.

# Appendix D: Additional information on AMT experiment

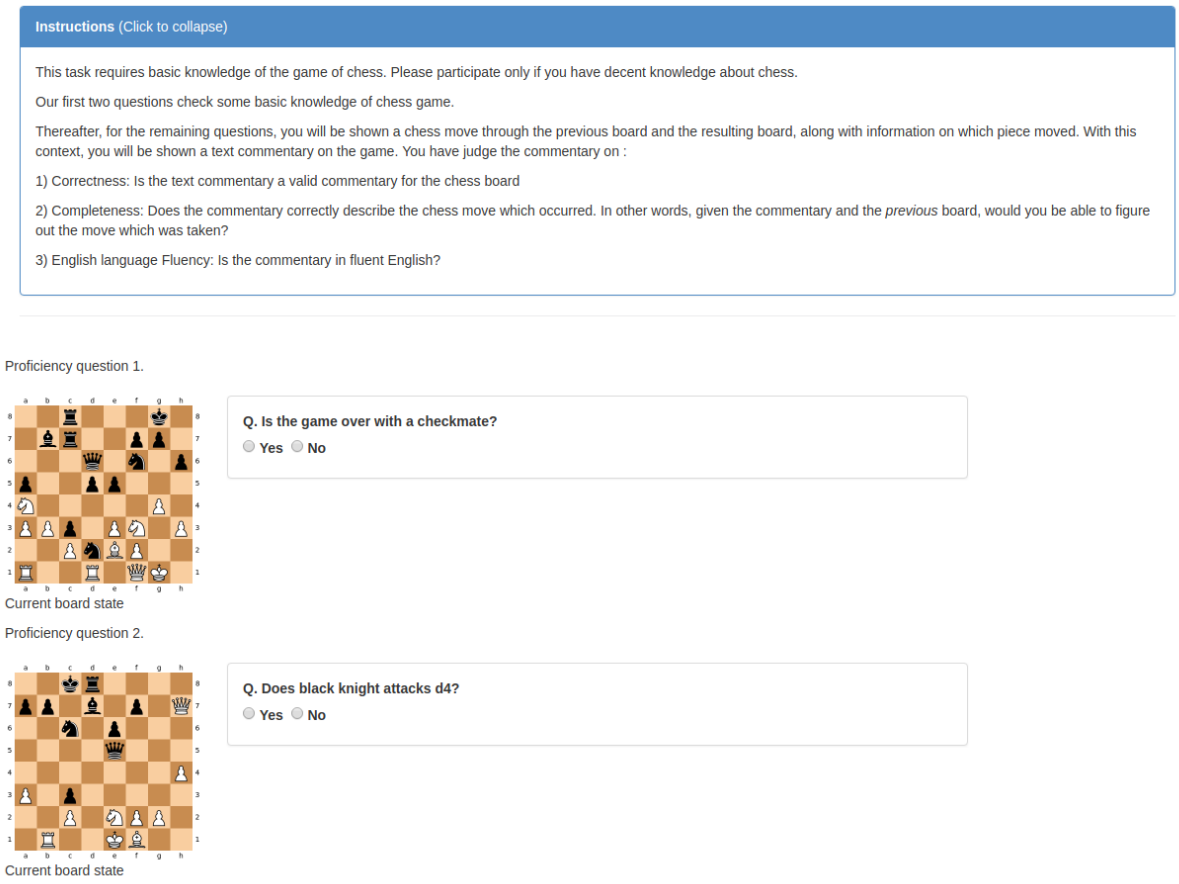



Figure 7.23: AMT (Amazon Mechanical Turk) sample HIT (Human Intelligence Task): Part 1 of 2 : Two chess proficiency questions are asked at beginning of a HIT

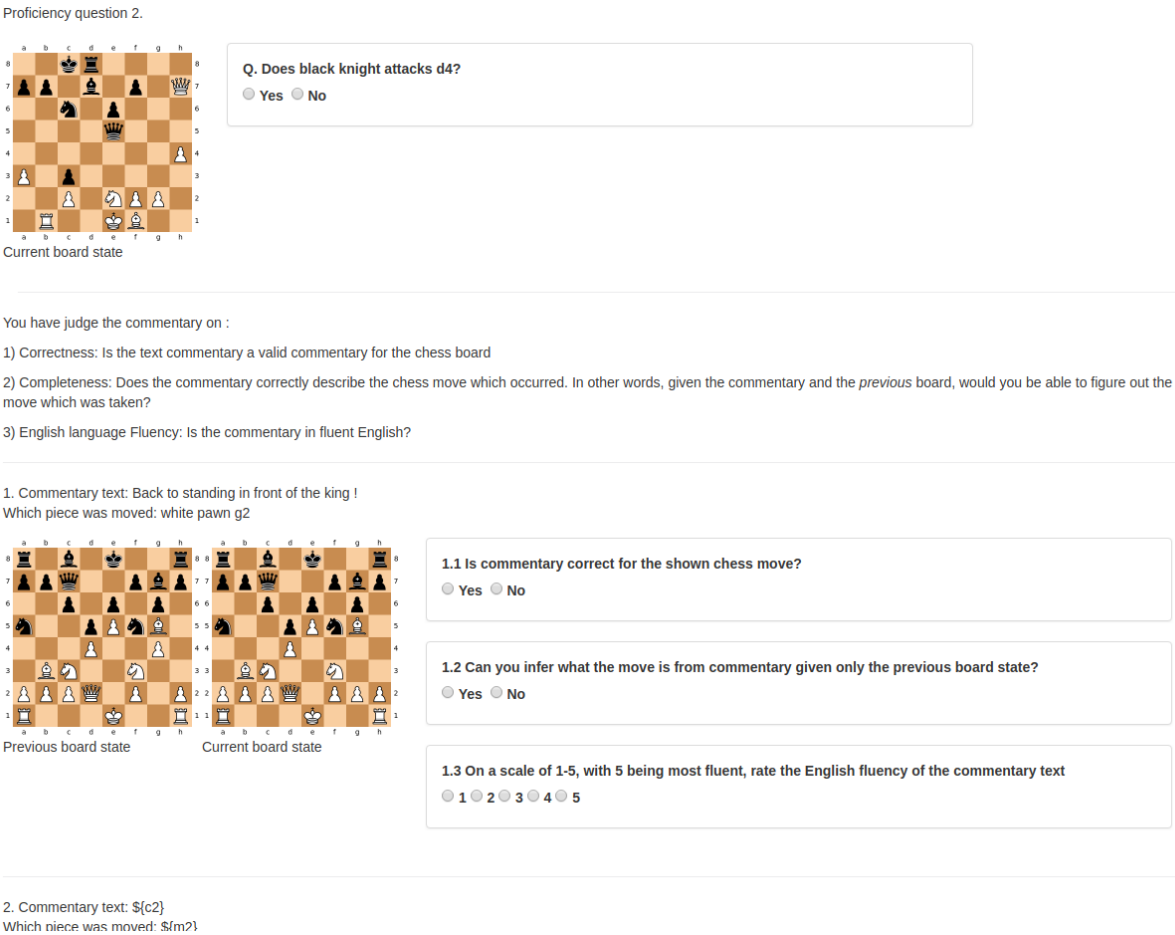



Figure 7.24: AMT (Amazon Mechanical Turk) sample HIT (Human Intelligence Task): Part 2 of 2: 7 sets of questions are asked to judge quality of generated text. Each of the seven texts is output from a different method.

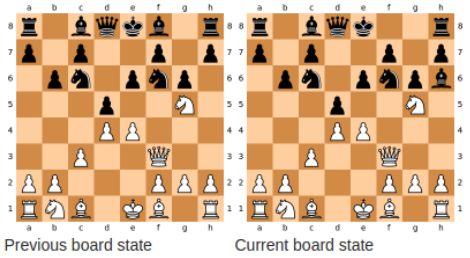



Figure 7.25: Commentary text: *I develop my bishop to the queen* .
An example instance where output commentary from our method was marked as not valid for the given chess move

## Checking chess proficiency of annotators

Our proficiency test questions are chosen from a subset of questions by [24]. Each question consists of a chess board and a question about the board configuration or game situation. The paper formulates a range of question types such as enumerating pieces of a type, enumerating pieces of a player, whether one piece threatens another, and whether the configuration corresponds to a checkmate or stalemate. For simplicity we stick to only those question types that have binary answer response.

We classify the question types into **Easy** and **Hard** question types. Each annotator is presented with one **Easy** and one **Hard** question at the start of a HIT.

# Part III

# Conclusion

July 24,2022

# Chapter 8

# Conclusion

In this thesis, we identified and characterized two distinct classes of data-deficient generation settings that present challenges for learning suitable end-to-end NLG models.

This thesis described a diverse set of setting-inspired formulations such as pretraining schemes (Chapters 2 and 5), model refactoring (Chapter 2), new architectural components (Chapter 5), data imputation (Chapters 3 and 4), unsupervised constraint incorporating procedures (Chapter 5) and input augmentation (Chapter 6). In each of the six NLG settings investigated, we posited that the respective interventions performed to the E2EN2PPs based on one or more of these formulations would lead to a final, improved NLG system that viably generate outputs sufficiently satisfying the CG. For each setting, the ensuing experiments confirmed this hypothesis, showing the post-intervention NLG system produced outputs which conformed better to the CG compared to the initial, baseline NLG system which instantiated the E2EN2PP sans any interventions.

## 8.1 Summary of Contributions

This thesis makes the following key contributions:

1. Commencing Part 1.3.1, Chapter 2 studies the setting of generating a portmanteau given its root words. We first sketch a creative story outlining how a human speaker would utilize two internal cognitive models based on root word predictability and English-word likeness. Using Bayesian refactoring, we intervene to modify the simple, character-level Seq2Seq model $P(y|x)$ (FORWARD) into a noisy channel style model $P(x|y)P(y)$ (BACKWARD), bringing the generation process closer to the creative story. We also show how the P(y) prior component introduced can be pretrained using the English vocabulary. Overall, BACKWARD outperforms both FORWARD as well as phonetic information dependent automata-based baselines from

prior art [30], underscoring the efficacy of creative story-driven interventions to the E2EN2PP. Additionally, this chapter also contributed a much-expanded corpus for studying portmanteau generation, 3 times larger than the existing corpus.

2. Continuing Part 1.3.1, Chapter 3 studies introducing personification into a source sentence not exhibiting it already, by assigning an inanimate entity animacy-requiring attributes/roles. This setting suffers from incomplete individual training examples, due to absence of parallel depersonified input and personified output sentence pairs in the wild. We first sketch out a creative story based on inanimate TOPIC and a animacy-requiring dependent ATTRIBUTE derived from the dependency structure, motivated by similar theories underlying metaphor formation. Based on this creative story, we devise a de-personification pipeline (see 1.1 for more) using off-the-shelf resources; thus constructing noisy, depersonified inputs to facilitate training. Effectively, this is an intervention to the E2EN2PP based on data imputation.
   Additionally, this chapter also introduced a novel generation setting i.e., personification generation and an associated corpus of $\approx$ 350 personified sentences. This setting requires thinking about dependency structure underlying the generated sentence and the commonsense-derived relations and constraints between the parts related through this structure.
3. Continuing Part 1.3.1, Chapter 4 introduces a novel phonetics-aware generation setting i.e., tongue twister generation and an associated corpus. Tongue twisters are defined as sentences that are meaningful but articulatorily difficult. Besides the data deficiency (only $\approx$ 400 examples), an added challenge to surmount is to finetune pretrained NLG models to sample difficult transitions in phoneme space while simultaneously maintaining meaningfulness in grapheme space. Overcoming these challenges, we devise an intervention to the learning process in the form of a novel, heterogenous training mechanism which instead of finetuning the model in grapheme-to-grapheme (G2G) mode alone, as would be typical, heterogenously finetunes it to generate either of phoneme/grapheme completions from prompts in either of phoneme/grapheme form.
4. Concluding Part 1.3.1, Chapter 5 studies the setting of diachronically style transferring modern English sentences to Shakespearean English, given limited parallel data and a noisy, sparse lexicon of $\approx$ 1000 modern $\rightarrow$ Shakespearean word correspondences. We devise a suite of beneficial interventions exploiting the property of a shared language and the underlying lexico-syntactic nature of the style transfer. These include i) sharing the source and target word embeddings, ii) introducing a copy component, and most critically iii) Incorporating the pairwise lexicon by pre-tuning word representations leveraging the retrofitting algorithm [39].
5. Commencing Part 1.3.2 , Chapter studies the Commongen [99] setting, where the CG is

to generate a plausible situation-sentence from a given set of input concepts. We posit that properties specific to the textual modality, e.g., the Gricean maxim of Quantity and Zipfian nature of concept occurrence, have a marked negative downstream effect on the NLG model's learning for CommonGen, leading to model generated outputs with poor plausibility, inadequate lexical relationships, and incomplete arguments etc. We devise an intervention through a novel method to symbolically augment input concepts by drawing information from the visual modality via a retrieve-and-caption pipeline, to help dampen this negative effect.

6. Concluding Part 1.3.2 , Chapter 7 introduces the novel setting of generating a short, interesting commentary sentence for each chess game move during gameplay. We show how S2S models simply based on a E2EN2PP fail to produce commentary that is even at the level of a template based baseline. We posit that this arises from inability to acquire tabula rasa the pragmatic knowledge necessary to understand the game situation. We devise a knowledge-incorporating intervention that includes an additional "Pragmatic Interpretation" layer to discretely featurize board states using the pychess library, backing off to pragmatic knowledge to better represent input states, simplifying their understanding and overcoming microplanning and macroplanning issues observed in the earlier output. This chapter also contributes a publically available corpus for this setting to facilitate further research by the community.

## 8.2 Limitations

### 8.2.1 Variation in Language and Dialect

In each of the six settings studied in this thesis, we primarily attempted to generate text that was contemporary American English, except when the setting specification itself required otherwise. As recommended by the Bender rule [9], we acknowledge that some of our findings may be specific to English, particularly the dominant American dialect. A natural point of inquiry for future work, would hence be: How well does our hypothesis about these classes of data-deficient settings generalize to other languages?

### 8.2.2 Use of Automatic, Reference-Based Metrics

At several points in this thesis, e.g., Chapter 5 , we employ reference-based evaluation using BLEU or other automatic metrics (such as BERTScore [191] with the reference) both as the primary form of evaluation during model development and an accompaniement to human evaluation during

final comparison of approaches and architechtures.

We acknowledge that such reference-based automatic evaluation is problematic in several regards [120], such as sensitivity to paraphrasing and the coverage and quality of references available [46]. Acquiring a high-quality, high-coverage set of references is particularly difficult for tasks with a creative aspect such as dialog, story generation and the settings we study in this thesis. This is because such tasks have a large space of valid responses.

Nevertheless, human evaluation is costly in terms of both money and time, besides being hard to reproduce. This leaves us with limited choice but to fall back on automatic reference-based metrics, particularly during model development. A promising alternative for future investigation are the recently emerging family of model-based, referenceless metrics, which we discuss further in §8.3.2.

## 8.3 Future Directions

In this section, we outline possible directions to extend our work in this thesis. These directions go beyond those already hinted at in the individual Future Work sections concluding each of our earlier chapters. We will first enlist some directions specifically extending some chapter in §8.3.1, followed by some broader ones in §8.3.2.

### 8.3.1 Specific Directions

#### Portmanteau Generation From Concept Description

The setting we studied in Chapter 2 had the CG of generating a portmanteau given its two root words. However, for practical applications of this setting, such as for brand and product nomenclature, assuming the user would know the root words beforehand is not realistic. Typically, the user would only be able to provide a single, terse sentence describing the concept for which they want to construct a portmanteau e.g., *A multipurpose eating utensil with prongs as well as curved spoonlike bowl at its tip.* (A potential/likely output here being *spork*)

How well would our intervention from Chapter 2 generalize to this setting? How would we need to update the underlying creative story to work for each this setting?

#### Multi-Target, Few-Shot Style Transfer

The setting we studied in Chapter 5 had the CG of style transferring a source text in modern English to Shakespearean English whilst preserving meaning. For actual applications, user needs may require a setting where it is possible to transfer to a large number of target styles e.g., Shakespeare, Milton and Shelley, with the choice amongst them being controlled by the user at test-time. We posit that this setting is likely to be more challenging on two aspects:

1. **A Harder Learning Problem:** Given the same parameter size, learning to transfer to multiple target styles requires learning a family of transduction functions rather than just a single one. The more complex function to be learnt makes this a harder learning problem.
2. **Greater Data Deficiency per Style:** It would not be realistic to expect the overall number of training examples to increase. As a conequence, we expect to find fewer examples per target style than before.

### 8.3.2 Broader Directions

#### Devising Automatic Metrics for Better Evaluation

Recent work has seen the formulation of several, task-specific “model-based" metrics, such as UNION [57] for story generation. These metrics contrastively train a model to score references higher than negative response sets, with the latter being constructed heuristically using task-specific procedures. For instance, UNION constructs negative responses through reordering sentences, as well as randomly performing N-gram and sentence-level repetition and substitution.

Once trained, these metrics are capable of performing reference-free automatic evaluation on test examples based on test examples alone.

However, since these metrics require task-specific strategies to construct the negative set of responses for training them, they are not readily extensible to new settings like those in this thesis. Moreover, training the models underlying these metrics is likely to be challenging for the data-deficient settings we study. Devising and training such metrics for each of the settings herein, and subsequently revisiting the experiments, remains a point for future investigation.

#### Discovering Intersecting Settings

The two classes of data-deficient settings we identify i.e., Constrained Creative settings, where the CG places esoteric constraints on the output, and Knowledge-Deficient settings, where the CG is underspecified w.r.t. the relationship it requires between output and input are not

mutually exclusive in theory. This is since a CG could satisfy both the defining characteristics simultaneously.

One natural question that hence arises is whether one can construct, identify and study NLG settings which belong to both these classes. Solving such settings could potentially be more challenging, since they would require devising interventions that not only incorporate an underlying creative story, but also incorporating one or more knowledge resources or other means to bridge the knowledge gap.

An example of such a NLG setting is generating a sarcastic comment about a given story. This setting is data-deficient since ground truth pairs of stories and sarcastic comments are hard to curate. This is a Constrained Creative setting since “being sarcastic" is a difficult constraint to satisfy, requiring generation of a sentence whose literal meaning violates the readers expectations, leading them to infer the opposite, sarcastic meaning. At the same time, this is also a Knowledge Deficient setting since the generated comment must also be interesting, adequate and plausible in the context of the input story.

**Using Other Settings Rather Than Incorporate Resources for Part II**

For the class of Knowledge Deficient Settings, the typical recipe we propose to construct an intervention involves incorporating a resource which can bridge the gap left by the CG and the deficient data w.r.t. the input-output relationship. For instance, the settings in Chapters 6 and 7 are viably intervened into using retrieved cross-modality information and python library information respectively.

However, another potential, intuitive way to bridge this knowledge gap could be by pretraining all or part of the NLG model on other related NLG settings, with the expectation being that the model would acquire the required knowledge in the process of being trained for the related task. For example, for the Commongen setting we study in Chapter 6, pretraining on the entailment generation NLG task i.e., generating an entailed hypothesis given a premise sentence, could potentially prove useful. Though this setting is not exactly the same as generating plausible situations from sets of concept words, it could provide an useful initial grounding on reasoning about objects and the plausible relationships/affordances they share with respect to each other.

Some natural, follow-up questions would be: Do these kind of conjugate setting pairs exhibit a common theme or pattern? Given any Knowledge Deficient NLG setting, can we easily identify suitable conjugate tasks to pretrain our representation?

**Discovering Entirely Newer Classes of Challenging, Data-Deficient NLG Settings**

The two classes of settings we study in Parts 1.3.1 and 1.3.2 are by no means exhaustive in their coverage of the space of NLG settings. A potential point for future investigation would be characterizing and identifying other such classes of settings where end-to-end NLG models based on the E2EN2PP do not directly work well.

# Bibliography

[1] Akintunde Akinyemi. Yorùbá oral literature: A source of indigenous education for children. *Journal of African Cultural Studies*, 16(2):161–179, 2003. 4.1

[2] John Algeo. Blends, a structural and systemic view. *American speech*, 52(1/2):47–64, 1977. 2.1

[3] Peter Anderson, Basura Fernando, Mark Johnson, and Stephen Gould. Spice: Semantic propositional image caption evaluation. In *European conference on computer vision*, pages 382–398. Springer, 2016. 6.2.3

[4] Peter Anderson, Xiaodong He, Chris Buehler, Damien Teney, Mark Johnson, Stephen Gould, and Lei Zhang. Bottom-up and top-down attention for image captioning and visual question answering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018. 6.11

[5] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv:1409.0473*, 2014. 3, 2.3.1, 5, 5.9, 5.10

[6] Tadas Baltrusaitis, Chaitanya Ahuja, and Louis-Philippe Morency. Multimodal machine learning: A survey and taxonomy. *IEEE Trans. Pattern Anal. Mach. Intell.*, 41(2):423–443, February 2019. ISSN 0162-8828. 6.7

[7] Outi Bat-El. Selecting the best of the worst: the grammar of Hebrew blends. *Phonology*, 13(03):283–328, 1996. 2.1

[8] Yonatan Belinkov, Ahmed M. Ali, and James R. Glass. Analyzing phonetic and graphemic representations in end-to-end automatic speech recognition. *ArXiv*, abs/1907.04224, 2019. 4.6

[9] Emily M Bender and Batya Friedman. Data statements for natural language processing: Toward mitigating system bias and enabling better science. *Transactions of the Association for Computational Linguistics*, 6:587–604, 2018. 8.2.1

[10] Ruth Berman. The role of blends in Modern Hebrew word-formation. *Studia linguistica et orientalia memoriae Haim Blanc dedicata. Wiesbaden: Harrassowitz*, pages 45–61, 1989. 2.1

[11] Julia Birke and Anoop Sarkar. Active learning for the identification of nonliteral language. In *Proceedings of the Workshop on Computational Approaches to Figurative Language*, pages 21–28, Rochester, New York, April 2007. Association for Computational Linguistics. 3.5

[12] Morton W Bloomfield. Personification-metaphors. *The Chaucer Review*, pages 287–297, 1980. 3.1

[13] Antoine Bosselut, Hannah Rashkin, Maarten Sap, Chaitanya Malaviya, Asli Çelikyilmaz, and Yejin Choi. COMET: commonsense Transformers for Automatic Knowledge Graph Construction. In Anna Korhonen, David R. Traum, and Lluís Màrquez, editors, *ACL*, 2019. 6.1

[14] Antoine Bosselut, Hannah Rashkin, Maarten Sap, Chaitanya Malaviya, Asli Celikyilmaz, and Yejin Choi. COMET: Commonsense transformers for automatic knowledge graph construction. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4762–4779, Florence, Italy, July 2019. Association for Computational Linguistics. 3.2.2

[15] Leo Breiman. Bagging predictors. *Machine learning*, 24(2):123–140, 1996. 2.4

[16] Roger Brown, Albert Gilman, et al. The pronouns of power and solidarity. *Article*, 1960. 5.1

[17] CJ Carey. Twisty tongue. `https://github.com/perimosocordiae/twisty_tongue`, 2017. 4.6

[18] Tuhin Chakrabarty, Smaranda Muresan, and Nanyun Peng. Generating similes effortlessly like a pro: A style transfer approach for simile generation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6455–6469, Online, November 2020. Association for Computational Linguistics. 3.1, 3.2.1, 3.5, 4.6

[19] Tuhin Chakrabarty, Xurui Zhang, Smaranda Muresan, and Nanyun Peng. MERMAID: Metaphor generation with symbolism and discriminative decoding. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4250–4261, Online, June 2021.

Association for Computational Linguistics. 3.1, 3.5, 4.6

[20] David L Chen and William B Dolan. Collecting highly parallel data for paraphrase evaluation. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 190–200. Association for Computational Linguistics, 2011. 5.7.3

[21] David L Chen and Raymond J Mooney. Learning to sportscast: a test of grounded language acquisition. In *Proceedings of the 25th international conference on Machine learning*, pages 128–135. ACM, 2008. 7.5

[22] Jianfu Chen, Polina Kuznetsova, David Warren, and Yejin Choi. Déjà image-captions: A corpus of expressive descriptions in repetition. In *HLT-NAACL*, pages 504–514, 2015. 3.2

[23] Junyoung Chung, Kyunghyun Cho, and Yoshua Bengio. A character-level decoder without explicit segmentation for neural machine translation. *arXiv:1603.06147*, 2016. 2.2

[24] Volkan Cirik, Louis-Philippe Morency, and Eduard Hovy. Chess q&a: Question Answering on Chess Games. In *Reasoning, Attention, Memory (RAM) Workshop, Neural Information Processing Systems*, 2015. 7.4.5, 7.6.3

[25] José Coch. Interactive generation and knowledge administration in multimeteo. In *Proc. 9th International Workshop on Natural Language Generation (INLG-98), Aug.*, 1998. 7.5

[26] Jacob Cohen. Weighted kappa: Nominal scale agreement provision for scaled disagreement or partial credit. *Psychological bulletin*, 70(4):213, 1968. 7.4.5

[27] Eric Corlett and Gerald Penn. An exact A* method for deciphering letter-substitution ciphers. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 1040–1047, Uppsala, Sweden, July 2010. Association for Computational Linguistics. 4.6

[28] T. V. D. Cruys. Automatic poetry generation from prosaic text. In *ACL*, 2020. 4.6

[29] Soenjono Dardjowidjojo. Acronymic Patterns in Indonesian. *Pacific Linguistics Series C*, 45:143–160, 1979. 2.1

[30] Aliya Deri and Kevin Knight. How to make a frenemy: Multitape FSTs for portmanteau generation. In *Proceedings of NAACL-HLT*, pages 206–210, 2015. (document), 2, 2.1, 2.1, 2.5, 2.6, 2.3, 2.8, 2.8.1, 4, 2.9, 1

[31] Tim Dettmers, Mike Lewis, Sam Shleifer, and Luke Zettlemoyer. 8-bit optimizers via block-wise quantization, 2021. 4.4.1

[32] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018. 6.2.3

[33] Aletta G Dorst. Personification in discourse: Linguistic forms, conceptual structures and communicative functions. *Language and Literature*, 20(2):113–135, 2011. 3.1

[34] Wenchao Du and Alan W Black. Boosting dialog response generation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 38–43, Florence, Italy, July 2019. Association for Computational Linguistics. 1.3.2

[35] Wenchao Du and Alan W Black. Boosting dialog response generation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 38–43, 2019. 6.3.1, 7

[36] Ondřej Dušek, Jekaterina Novikova, and Verena Rieser. Findings of the E2E NLG challenge. In *Proceedings of the 11th International Conference on Natural Language Generation*, pages 322–328, Tilburg University, The Netherlands, November 2018. Association for Computational Linguistics. 6.7

[37] Rodney Stenning Edgecombe. Ways of personifying. *Style*, 31(1):1–13, 1997. ISSN 00394238. 3.5

[38] Zhihao Fan, Yeyun Gong, Zhongyu Wei, Siyuan Wang, Yameng Huang, Jian Jiao, Xuanjing Huang, Nan Duan, and Ruofei Zhang. An enhanced knowledge injection model for commonsense generation. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 2014–2025, Barcelona, Spain (Online), December 2020. International Committee on Computational Linguistics. 6.6.1, **??**, 6.7

[39] Manaal Faruqui, Jesse Dodge, Sujay K Jauhar, Chris Dyer, Eduard Hovy, and Noah A Smith. Retrofitting word vectors to semantic lexicons. *arXiv preprint arXiv:1411.4166*, 2014. **??**, 5.1.2, 5.4.1, 5.10, 4

[40] Manaal Faruqui, Yulia Tsvetkov, Graham Neubig, and Chris Dyer. Morphological Inflection Generation using Character Sequence to Sequence Learning. In *Proceedings of NAACL-HLT*, pages 634–643, 2016. 2.1

[41] Christiane Fellbaum. *WordNet: An Electronic Lexical Database*. Bradford Books, 1998. 4.2.2

[42] Steven Y Feng, Aaron W Li, and Jesse Hoey. Keep Calm and Switch On! Preserving

Sentiment and Fluency in Semantic Text Exchange. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2701–2711, 2019. 6.7

[43] Steven Y Feng, Kevin Lu, Zhuofu Tao, Malihe Alikhani, Teruko Mitamura, Eduard Hovy, and Varun Gangal. Retrieve, caption, generate: Visual grounding for enhancing commonsense in text generation models. *arXiv preprint arXiv:2109.03892*, 2021. 1.3.2

[44] Steven Y Feng, Kevin Lu, Zhuofu Tao, Malihe Alikhani, Teruko Mitamura, Eduard Hovy, and Varun Gangal. Retrieve, caption, generate: Visual grounding for enhancing commonsense in text generation models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 10618–10626, 2022. 2

[45] Mary C Flannery. Personification and embodied emotional practice in middle english literature. *Literature Compass*, 13(6):351–361, 2016. 3.1

[46] Markus Freitag, David Grangier, and Isaac Caswell. Bleu might be Guilty but References are not Innocent. *arXiv preprint arXiv:2004.06063*, 2020. 8.2.2

[47] Neal Gabler. The Weird Science of Naming New Products. *New York Times - http://tinyurl.com/lmlq7ex*, 1 2015. 2.1

[48] Varun Gangal, Harsh Jhamtani, Graham Neubig, Eduard Hovy, and Eric Nyberg. Charmanteau: Character embedding models for portmanteau creation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2917–2922, Copenhagen, Denmark, September 2017. Association for Computational Linguistics. 1.3.1

[49] Varun Gangal, Harsh Jhamtani, Graham Neubig, Eduard Hovy, and Eric Nyberg. Charmanteau: Character embedding models for portmanteau creation. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Copenhagen, Denmark, September 2017. 5.9

[50] Ge Gao, Eunsol Choi, Yejin Choi, and Luke Zettlemoyer. Neural metaphor detection in context. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 607–613, Brussels, Belgium, October-November 2018. Association for Computational Linguistics. 3.5

[51] Jing Gao, Peng Li, Zhikui Chen, and Jianing Zhang. A Survey on Deep Learning for Multimodal Data Fusion. *Neural Computation*, 32(5):829–864, 05 2020. ISSN 0899-7667. 6.7

[52] Claire Gardent, Anastasia Shimorina, Shashi Narayan, and Laura Perez-Beltrachini. The webnlg challenge: Generating text from rdf data. In *Proceedings of the 10th International Conference on Natural Language Generation*, pages 124–133, 2017. 1.2.2, 6.7

[53] Andrew Gargett and John Barnden. Gen-meta: Generating metaphors using a combination of ai reasoning and corpus-based modeling of formulaic expressions. pages 103–108, 12 2013. 3.5

[54] Marjan Ghazvininejad, Xing Shi, Jay Priyadarshi, and Kevin Knight. Hafez: an interactive poetry generation system. pages 43–48, 01 2017. 4.6

[55] Jonathan Gordon and Benjamin Van Durme. Reporting bias and knowledge acquisition. In *Proceedings of the 2013 workshop on Automated knowledge base construction*, pages 25–30, 2013. 6, 6.1, 6.3.2

[56] Maarten Grootendorst. Keybert: Minimal keyword extraction with bert., 2020. 2, 4.4.1

[57] Jian Guan and Minlie Huang. Union: an unreferenced metric for evaluating open-ended story generation. *arXiv preprint arXiv:2009.07602*, 2020. 8.3.2

[58] M. A. K. Halliday. Systemic functional linguistics: Exploring choice: Meaning as choice. In *Systemic Functional Linguistics: Exploring Choice: Meaning as choice*, 2013. 5

[59] Craig A. Hamilton. Mapping the mind and the body: On w. h. auden's personifications. *Style*, 36(3):408–427, 2002. ISSN 00394238, 23746629. 3.5

[60] Farzaneh Haratyan. Halliday's sfl and social meaning. In *2nd International Conference on Humanities, Historical and Social Sciences*, volume 17, pages 260–264, 2011. (document), 1.5

[61] Devamanyu Hazarika, Soujanya Poria, Sruthi Gorantla, Erik Cambria, Roger Zimmermann, and Rada Mihalcea. CASCADE: Contextual sarcasm detection in online discussion forums. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 1837–1848, Santa Fe, New Mexico, USA, August 2018. Association for Computational Linguistics. 3.1

[62] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, pages 770–778. IEEE Computer Society, 2016. 6.11

[63] Ilana Heintz, Ryan Gabbard, Mahesh Srivastava, Dave Barner, Donald Black, Majorie

Friedman, and Ralph Weischedel. Automatic extraction of linguistic metaphors with LDA topic modeling. In *Proceedings of the First Workshop on Metaphor in NLP*, pages 58–66, Atlanta, Georgia, June 2013. Association for Computational Linguistics. 3.5

[64] Gaurush Hiranandani, Pranav Maneriker, and Harsh Jhamtani. Generating appealing brand names. *arXiv preprint arXiv:1706.09335*, 2017. 2.2, 5.1

[65] Cong Duy Vu Hoang, Gholamreza Haffari, and Trevor Cohn. Decoding as Continuous Optimization in Neural Machine Translation. *arXiv:1701.02854*, 2017. 2.3.2

[66] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997. 7.3

[67] Jack Hopkins and Douwe Kiela. Automatically generating rhythmic verse with neural networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 168–178, Vancouver, Canada, July 2017. Association for Computational Linguistics. 4.1.2, 4.6

[68] Dirk Hovy, Shashank Srivastava, Sujay Kumar Jauhar, Mrinmaya Sachan, Kartik Goyal, Huying Li, Whitney Sanders, and Eduard Hovy. Identifying metaphorical word use with tree kernels. In *Proceedings of the First Workshop on Metaphor in NLP*, pages 52–57, Atlanta, Georgia, June 2013. Association for Computational Linguistics. 3.5

[69] Eduard H Hovy. Pragmatics and natural language generation. *Artificial Intelligence*, 43(2): 153–197, 1990. (document), 4, 1.5

[70] Edward Hu, Yelong Shen, Phil Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models, 2021. 4.4.1

[71] Nikhil Jaiswal. Neural sarcasm detection using conversation context. In *Proceedings of the Second Workshop on Figurative Language Processing*, pages 77–82, Online, July 2020. Association for Computational Linguistics. 3.1

[72] Bashir Jam. Features of tongue twisters and analysis of their structure in persian. *Journal of Researches in Linguistics*, 9(2):59–76, 2018. 4.1

[73] Hyeju Jang, Seungwhan Moon, Yohan Jo, and Carolyn Rosé. Metaphor detection in discourse. In *Proceedings of the 16th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 384–392, Prague, Czech Republic, September 2015. Association for Computational Linguistics. 3.5

[74] Harsh Jhamtani, Varun Gangal, Eduard Hovy, and Eric Nyberg. Shakespearizing

Modern language using Copy-Enriched Sequence-to-Sequence Models. *arXiv preprint arXiv:1707.01161*, 2017. 1.3.1

[75] Harsh Jhamtani, Varun Gangal, Eduard Hovy, Graham Neubig, and Taylor Berg-Kirkpatrick. Learning to generate move-by-move commentary for chess games from large-scale social forum data. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1661–1671, 2018. 1.3.2

[76] Melvin Johnson, Mike Schuster, Quoc V Le, Maxim Krikun, Yonghui Wu, Zhifeng Chen, Nikhil Thorat, Fernanda Viegas, Martin Wattenberg, Greg Corrado, et al. Google's multilingual neural machine translation system: Enabling zero-shot translation. *arXiv preprint arXiv:1611.04558*, 2016. 5.10

[77] Marilyn Jorgensen. The tickled, tangled, tripped, and twisted tongue: A linguistic study of factors relating to difficulty in the performance of tongue twisters. *New York Folklore*, 7(3): 67, 1981. 4.1.2

[78] Dweep Joshipura. Tongue twister generation. https://www.kaggle.com/code/djathidiro/tongue-twister-generation/notebook, 2020. 4.6

[79] Hirotaka Kameko, Shinsuke Mori, and Yoshimasa Tsuruoka. Learning a game commentary generator with grounded move expressions. In *Computational Intelligence and Games (CIG), 2015 IEEE Conference on*, pages 177–184. IEEE, 2015. 7.1

[80] Andrej Karpathy and Li Fei-Fei. Deep visual-semantic alignments for generating image descriptions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3128–3137, 2015. 7.1

[81] Heather Kember, Kathryn Connaghan, and Rupal Patel. Inducing speech errors in dysarthria using tongue twisters. *International journal of language & communication disorders*, 52 (4):469–478, 2017. 4.1

[82] Maurice G Kendall. A new measure of rank correlation. *Biometrika*, 30(1/2):81–93, 1938. 7.4.5

[83] Chloé Kiddon, Luke Zettlemoyer, and Yejin Choi. Globally Coherent Text Generation with Neural Checklist Models. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 329–339, 2016. 7.5

[84] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference on Learning*

*Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015. 5.7.4, 7.4

[85] Tibor Kiss and Jan Strunk. Unsupervised multilingual sentence boundary detection. *Computational linguistics*, 32(4):485–525, 2006. 5.7.1

[86] Eliza Kitis. Ads—part of our lives: linguistic awareness of powerful advertising. *Word & Image*, 13(3):304–313, 1997. 5.1

[87] Kevin Knight, Anish Nair, Nishit Rathod, and Kenji Yamada. Unsupervised analysis for decipherment problems. In *Proceedings of the COLING/ACL 2006 Main Conference Poster Sessions*, pages 499–506, Sydney, Australia, July 2006. Association for Computational Linguistics. 4.6

[88] Wei-Jen Ko and Junyi Jessy Li. Assessing discourse relations in language generation from GPT-2. In *Proceedings of the 13th International Conference on Natural Language Generation*, pages 52–59, Dublin, Ireland, December 2020. Association for Computational Linguistics. (document), 1.1

[89] Philipp Koehn. Statistical significance tests for machine translation evaluation. In *EMNLP*, pages 388–395, 2004. 2.8.2

[90] Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, et al. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th annual meeting of the ACL on interactive poster and demonstration sessions*, pages 177–180. Association for Computational Linguistics, 2007. 5.7.2

[91] Ranjay Krishna, Yuke Zhu, O. Groth, Justin Johnson, K. Hata, J. Kravitz, Stephanie Chen, Yannis Kalantidis, Li-Jia Li, D. Shamma, Michael S. Bernstein, and Li Fei-Fei. Visual genome: Connecting language and vision using crowdsourced dense image annotations. *International Journal of Computer Vision*, 123:32–73, 2016. 6.11

[92] Peter Ladefoged and Keith Johnson. *A course in phonetics*. Cengage learning, 2014. 1.3.1, 4.1.1

[93] Rémi Lebret, David Grangier, and Michael Auli. Neural text generation from structured data with application to the biography domain. *arXiv preprint arXiv:1603.07771*, 2016. 7.1, 7.5

[94] Vladimir I Levenshtein et al. Binary codes capable of correcting deletions, insertions, and

reversals. In *Soviet physics doklady*, volume 10, pages 707–710. Soviet Union, 1966. 2

[95] Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Ves Stoyanov, and Luke Zettlemoyer. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *arXiv preprint arXiv:1910.13461*, 2019. 6.1

[96] Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 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, Online, July 2020. Association for Computational Linguistics. 3.2.2

[97] Percy Liang, Michael I Jordan, and Dan Klein. Learning semantic correspondences with less supervision. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 1-Volume 1*, pages 91–99. Association for Computational Linguistics, 2009. 7.5

[98] Jen-Wen Liao and Jason S Chang. Computer Generation of Chinese Commentary on Othello Games. In *Proceedings of Rocling III Computational Linguistics Conference III*, pages 393–415, 1990. 7.1

[99] Bill Yuchen Lin, Wangchunshu Zhou, Ming Shen, Pei Zhou, Chandra Bhagavatula, Yejin Choi, and Xiang Ren. CommonGen: A constrained text generation challenge for generative commonsense reasoning. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 1823–1840, Online, November 2020. Association for Computational Linguistics. (document), 2, 1.1, 1.2.2, 1.3.2, 1.3.2, 6, 6.1, 6.2.1, 6.2.2, 6.2.3, 6.3, 6.6.1, 6.8, 6.8, 6.10, 6.13, 6.12, 5

[100] Chin-Yew Lin and Eduard Hovy. Automatic evaluation of summaries using n-gram co-occurrence statistics. In *Proceedings of the 2003 Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics*, pages 150–157, 2003. 6.2.2

[101] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014. 7.1

[102] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan,

Piotr Dollár, and C. Lawrence Zitnick. Microsoft coco: Common objects in context. In David Fleet, Tomas Pajdla, Bernt Schiele, and Tinne Tuytelaars, editors, *Computer Vision – ECCV 2014*, pages 740–755, Cham, 2014. Springer International Publishing. ISBN 978-3-319-10602-1. 6.1

[103] Wang Ling, Isabel Trancoso, Chris Dyer, and Alan W Black. Character-based neural machine translation. *arXiv:1511.04586*, 2015. 2.2

[104] Ye Liu, Yao Wan, Lifang He, Hao Peng, and Philip S. Yu. Kg-bart: Knowledge graph-augmented bart for generative commonsense reasoning. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(7):6418–6425, May 2021. 6.6.1, **??**, 6.7

[105] Deyin Long. Meaning construction of personification in discourse based on conceptual integration theory. *Studies in Literature and Language*, 17:21–28, 2018. 3.5

[106] Jiasen Lu, Dhruv Batra, Devi Parikh, and Stefan Lee. Vilbert: Pretraining task-agnostic visiolinguistic representations for vision-and-language tasks. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. 6.7

[107] Ruotian Luo, Brian Price, Scott Cohen, and Gregory Shakhnarovich. Discriminability objective for training descriptive captions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018. 6.4.2, 6.11

[108] Pranava Swaroop Madhyastha, Mohit Bansal, Kevin Gimpel, and Karen Livescu. Mapping unseen words to task-trained embedding spaces. *arXiv preprint arXiv:1510.02387*, 2015. 5.4.2

[109] Saad Mahamood and Ehud Reiter. Working with clinicians to improve a patient-information NLG system. In *Proceedings of the Seventh International Natural Language Generation Conference*, pages 100–104. Association for Computational Linguistics, 2012. 7.3.1

[110] Mitchell P Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. Building a large annotated corpus of english: The penn treebank. *Computational linguistics*, 19(2):313–330, 1993. 5.4.1

[111] Jonathan Mcgovern. Three Tudor Tongue-Twisters. *Notes and Queries*, 68(4):392–393, 12 2021. ISSN 0029-3970. 4.1

[112] Hongyuan Mei, Mohit Bansal, and Matthew R Walter. What to talk about and how? selective generation using LSTMs with coarse-to-fine alignment. *arXiv preprint*

*arXiv:1509.00838*, 2015. 7.5

[113] Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer Sentinel Mixture Models. *arXiv preprint arXiv:1609.07843*, 2016. 2, 5.1.1, 5.5.3, 5.6, 5.6, 5.9

[114] Ning Miao, Hao Zhou, Lili Mou, Rui Yan, and Lei Li. Cgmh: Constrained sentence generation by metropolis-hastings sampling. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 6834–6842, 2019. 6.7

[115] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013. 5.4.1

[116] Sebastien Montella, Betty Fabre, Tanguy Urvoy, Johannes Heinecke, and Lina Rojas-Barahona. Denoising pre-training and data augmentation strategies for enhanced RDF verbalization with transformers. In *Proceedings of the 3rd International Workshop on Natural Language Generation from the Semantic Web (WebNLG+)*, pages 89–99, Dublin, Ireland (Virtual), 12 2020. Association for Computational Linguistics. 6.7

[117] Aakanksha Naik. *Domain Adaptation for the Long Tail in Language Understanding*. PhD thesis, Carnegie Mellon University Pittsburgh, PA, 2022. 1.3.1, 4, 4.1

[118] Raymond J Nelson. The competence-performance distinction in mental philosophy. *Synthese*, pages 337–381, 1978. 1

[119] Vlad Niculae and Cristian Danescu-Niculescu-Mizil. Brighter than gold: Figurative language in user generated comparisons. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2008–2018, Doha, Qatar, October 2014. Association for Computational Linguistics. 3.2.1

[120] Jekaterina Novikova, Ondřej Dušek, Amanda Cercas Curry, and Verena Rieser. Why we need new evaluation metrics for nlg. *arXiv preprint arXiv:1707.06875*, 2017. 8.2.2

[121] Jekaterina Novikova, Ondřej Dušek, Amanda Cercas Curry, and Verena Rieser. Why we need new evaluation metrics for nlg. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2241–2252, Copenhagen, Denmark, September 2017. Association for Computational Linguistics. 7.4.2, 7.4.5

[122] Franz Josef Och. Minimum error rate training in statistical machine translation. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics-Volume 1*, pages 160–167. Association for Computational Linguistics, 2003. 5.7.2

[123] Gözde Özbal and Carlo Strapparava. A computational approach to the automation of

creative naming. In *Proceedings of ACL*, pages 703–711. Association for Computational Linguistics, 2012. 2.2

[124] Lalchand Pandia, Yan Cong, and Allyson Ettinger. Pragmatic competence of pre-trained language models through the lens of discourse connectives. *arXiv preprint arXiv:2109.12951*, 2021. (document), 1.1

[125] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pages 311–318. Association for Computational Linguistics, 2002. 1, 5.7.3, 6.2.3, 7.4

[126] Adam Paszke, Sam Gross, Soumith Chintala, and Gregory Chanan. Pytorch, 2017. 7.4

[127] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011. 7.2

[128] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12(Oct): 2825–2830, 2011. 7.4.1

[129] Nanyun Peng, Marjan Ghazvininejad, Jonathan May, and Kevin Knight. Towards controllable story generation. In *Proceedings of the First Workshop on Storytelling*, pages 43–49, 2018. 1.3.4

[130] Jeffrey Pennington, Richard Socher, and Christopher D Manning. Glove: Global Vectors for Word Representation. In *EMNLP*, volume 14, pages 1532–1543, 2014. 5.4.1

[131] Alexandra Petri. Say No to Portmanteaus. *Washington Post* - `http://tinyurl.com/kvmep2t`, 6 2012. 2.1

[132] Carlos-Eduardo Piñeros. The creation of portmanteaus in the extragrammatical morphology of Spanish. *Probus*, 16(2):203–240, 2004. 2.1

[133] Edwin JG Pitman. Significance tests which may be applied to samples from any populations. *Supplement to the Journal of the Royal Statistical Society*, 4(1):119–130, 1937. (document), 6.6, 6.7

[134] Shrimai Prabhumoye, Alan W Black, and Ruslan Salakhutdinov. Exploring controllable

text generation techniques. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 1–14, 2020. 1.3.4

[135] Danijela Prošić-Santovac. The use of tongue twisters in efl teaching. *Annual Review of the Faculty of Philosophy/Godisnjak Filozofskog Fakulteta*, 34, 2009. 4.1

[136] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. *OpenAI Blog*, 1(8):9, 2019. 3, 6.2.3

[137] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. *arXiv preprint arXiv:2103.00020*, 2021. 6.8.1

[138] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140): 1–67, 2020. 6.1

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

[140] Paul Rayson, Dawn Archer, Alistair Baron, Jonathan Culpeper, and Nicholas Smith. Tagging the Bard: Evaluating the accuracy of a modern POS tagger on Early Modern English corpora. *Article*, 2007. 5.1

[141] Ehud Reiter. NLG vs. templates. *arXiv preprint cmp-lg/9504013*, 1995. 7.4.1

[142] Ehud Reiter and Anja Belz. An investigation into the validity of some metrics for automatically evaluating natural language generation systems. *Computational Linguistics*, 35(4): 529–558, 2009. 7.4.5

[143] Ehud Reiter, Roma Robertson, and Liesl M Osman. Lessons from a failure: Generating tailored smoking cessation letters. *Artificial Intelligence*, 144(1-2):41–58, 2003. 7.5

[144] Ehud Reiter, Somayajulu G Sripada, and Roma Robertson. Acquiring correct knowledge for natural language generation. *Journal of Artificial Intelligence Research*, 18:491–516, 2003. 7.3.1

[145] Ehud Reiter, Somayajulu Sripada, Jim Hunter, Jin Yu, and Ian Davy. Choosing words in

computer-generated weather forecasts. *Artificial Intelligence*, 167(1-2):137–169, 2005. 7.5

[146] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc., 2015. 6.11

[147] Steven J. Rennie, Etienne Marcheret, Youssef Mroueh, Jerret Ross, and Vaibhava Goel. Self-critical sequence training for image captioning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017. 6.4.2, 6.11

[148] M Revathy and K Ravindran. Enhancing effective speaking skills through role play and tongue twisters. *Language in India*, 16(9), 2016. 4.1

[149] David E Rogers. The influence of panini on leonard bloomfield. *Historiographia linguistica*, 14(1-2):89–138, 1987. 1

[150] Ronald Rosenfeld. A whole sentence maximum entropy language model. In *1997 IEEE workshop on automatic speech recognition and understanding proceedings*, pages 230–237. IEEE, 1997. 2

[151] Alexander M Rush, Sumit Chopra, and Jason Weston. A neural attention model for abstractive sentence summarization. *arXiv preprint arXiv:1509.00685*, 2015. 5.1, 5.9

[152] Aleksander Sadikov, Martin Moina, Matej Guid, Jana Krivec, and Ivan Bratko. Automated chess tutor. In *International Conference on Computers and Games*, pages 13–25. Springer, 2006. 7.1

[153] Rishiraj Saha Roy, Aishwarya Padmakumar, Guna Prasaad Jeganathan, and Ponnurangam Kumaraguru. Automated Linguistic Personalization of Targeted Marketing Messages Mining User-Generated Text on Social Media. In *16th International Conference on Intelligent Text Processing and Computational Linguistics 2015 (CICLing '15)*, pages 203–224. Springer International Publishing, 2015. 5.1, 5.1.2, 5.9

[154] H Andrew Schwartz, Johannes C Eichstaedt, Margaret L Kern, Lukasz Dziurzynski, Stephanie M Ramones, Megha Agrawal, Achal Shah, Michal Kosinski, David Stillwell, Martin EP Seligman, et al. Personality, gender, and age in the language of social media: The open-vocabulary approach. *PloS one*, 8(9):e73791, 2013. 5.1

[155] Abigail See, Peter J Liu, and Christopher D Manning. Get To The Point: Summarization with Pointer-Generator Networks. *arXiv preprint arXiv:1704.04368*, 2017. 5.9

[156] Rico Sennrich, Barry Haddow, and Alexandra Birch. Controlling politeness in neural machine translation via side constraints. In *Proceedings of NAACL-HLT*, pages 35–40, 2016. 5.9

[157] Claude E Shannon. Prediction and entropy of printed English. *Bell Labs Technical Journal*, 30(1):50–64, 1951. 7.4.4

[158] Katherine E Shaw, Andrew M White, Elliott Moreton, and Fabian Monrose. Emergent faithfulness to morphological and semantic heads in lexical blends. In *Proceedings of the Annual Meetings on Phonology*, volume 1, 2014. 2.1, 2.1

[159] Vered Shwartz and Yejin Choi. Do neural language models overcome reporting bias? In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 6863–6870, 2020. 6.3.2

[160] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of Go with deep neural networks and tree search. *nature*, 529(7587):484–489, 2016. 7.6

[161] Aastha Singhal. Would You Interact with a Chatbot that's Unfriendly? 67% users say no! https://www.entrepreneur.com/article/339248, 2021. 5

[162] Michael R Smith, Ryan S Hintze, and Dan Ventura. Nehovah: A neologism creator nomen ipsum. In *Proceedings of the International Conference on Computational Creativity*, pages 173–181, 2014. 2.2

[163] Robyn Speer, Joshua Chin, and Catherine Havasi. Conceptnet 5.5: An open multilingual graph of general knowledge. In *AAAI*, 2017. 3.2.2

[164] Balaji Vasan Srinivasan, Rishiraj Saha Roy, Harsh Jhamtani, Natwar Modani, and Niyati Chhaya. Corpus-based automatic text expansion. In *CICLING*, 2017. 5.1

[165] VentureBeat Staff. People, not tech companies should pick their AI Assistant's personality. https://venturebeat.com/2017/10/, 2021. 5

[166] Kevin Stowe, Tuhin Chakrabarty, Nanyun Peng, Smaranda Muresan, and Iryna Gurevych. Metaphor generation with conceptual mappings. 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 6724–6736, 2021. 3.1, 3.5

[167] Tomek Strzalkowski, George Aaron Broadwell, Sarah Taylor, Laurie Feldman, Samira Shaikh, Ting Liu, Boris Yamrom, Kit Cho, Umit Boz, Ignacio Cases, and Kyle Elliot. Robust extraction of metaphor from novel data. In *Proceedings of the First Workshop on Metaphor in NLP*, pages 67–76, Atlanta, Georgia, June 2013. Association for Computational Linguistics. 3.5

[168] Mukuntha Narayanan Sundararaman, Ayush Kumar, and Jithendra Vepa. Phonemebert: Joint language modelling of phoneme sequence and asr transcript. *arXiv preprint arXiv:2102.00804*, 2021. 4.1.2, 4.6

[169] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. In *Neural information processing systems*, pages 3104–3112, 2014. 2.2, 5.9

[170] Swabha Swayamdipta, Sam Thomson, Kenton Lee, Luke Zettlemoyer, Chris Dyer, and Noah A Smith. Syntactic scaffolds for semantic structures. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3772–3782, 2018. 4

[171] Alon Talmor, Yanai Elazar, Yoav Goldberg, and Jonathan Berant. oLMpics-on what language model pre-training captures. *Transactions of the Association for Computational Linguistics*, 8:743–758, 2020. 6.1

[172] Yufei Tian, Arvind krishna Sridhar, and Nanyun Peng. HypoGen: Hyperbole generation with commonsense and counterfactual knowledge. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 1583–1593, Punta Cana, Dominican Republic, November 2021. Association for Computational Linguistics. 4.6

[173] Enrica Troiano, Carlo Strapparava, Gözde Özbal, and Serra Sinem Tekiroğlu. A computational exploration of exaggeration. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3296–3304, Brussels, Belgium, October-November 2018. Association for Computational Linguistics. 3.1

[174] Cynthia Van Hee, Els Lefever, and Véronique Hoste. SemEval-2018 task 3: Irony detection in English tweets. In *Proceedings of The 12th International Workshop on Semantic Evaluation*, pages 39–50, New Orleans, Louisiana, June 2018. Association for Computational Linguistics. 3.1

[175] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008, 2017. (document)

[176] Ramakrishna Vedantam, C Lawrence Zitnick, and Devi Parikh. Cider: Consensus-based image description evaluation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4566–4575, 2015. 6.2.3, 7.4

[177] Ashwin K Vijayakumar, Michael Cogswell, Ramprasath R Selvaraju, Qing Sun, Stefan Lee, David Crandall, and Dhruv Batra. Diverse beam search: Decoding diverse solutions from neural sequence models. *arXiv preprint arXiv:1610.02424*, 2016. 6.3.1, 7

[178] Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. Pointer networks. In *Advances in Neural Information Processing Systems*, pages 2692–2700, 2015. 5.9

[179] Han Wang, Yang Liu, Chenguang Zhu, Linjun Shou, Ming Gong, Yichong Xu, and Michael Zeng. Retrieval enhanced model for commonsense generation. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 3056–3062, Online, August 2021. Association for Computational Linguistics. 6.6.1, **??**

[180] Shuohang Wang and Jing Jiang. Machine comprehension using match-lstm and answer pointer. *arXiv preprint arXiv:1608.07905*, 2016. 5.9

[181] R Weide. The CMU pronunciation dictionary, release 0.6. *Carnegie Mellon University*, 1998. 2

[182] Sam Wiseman, Stuart M Shieber, and Alexander M Rush. Challenges in Data-to-Document Generation. *arXiv preprint arXiv:1707.08052*, 2017. 7.4.5, 7.5

[183] Qiantong Xu, Alexei Baevski, and Michael Auli. Simple and effective zero-shot cross-lingual phoneme recognition. *ArXiv*, abs/2109.11680, 2021. 4.6

[184] Wei Xu. *Data-driven approaches for paraphrasing across language variations*. PhD thesis, New York University, 2014. 5.2, 5.4.1

[185] Wei Xu, Alan Ritter, William B Dolan, Ralph Grishman, and Colin Cherry. Paraphrasing for style. In *24th International Conference on Computational Linguistics, COLING 2012*, 2012. 5.1.2, 5.2, 5.4.1, 5.7.2, 5.7.2, 5.9

[186] Zichao Yang, Phil Blunsom, Chris Dyer, and Wang Ling. Reference-Aware Language Models. *arXiv preprint arXiv:1611.01628*, 2016. 7.5

[187] Lei Yu, Phil Blunsom, Chris Dyer, Edward Grefenstette, and Tomas Kocisky. The Neural Noisy Channel. *arXiv:1611.02554*, 2016. 2.3.2

[188] Rowan Zellers, Yonatan Bisk, Ali Farhadi, and Yejin Choi. From recognition to cognition: Visual commonsense reasoning. In *Proceedings of the IEEE/CVF Conference on Computer*

*Vision and Pattern Recognition (CVPR)*, June 2019. 6.7

[189] Faith Zeng. Tongue twister generator, Apr 2019. 4.6

[190] Wenyuan Zeng, Wenjie Luo, Sanja Fidler, and Raquel Urtasun. Efficient Summarization with Read-Again and Copy Mechanism. *arXiv preprint arXiv:1611.03382*, 2016. 5.9

[191] Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q Weinberger, and Yoav Artzi. Bertscore: Evaluating text generation with bert. *arXiv preprint arXiv:1904.09675*, 2019. 2, 6.2.3, 8.2.2

[192] Tianyi Zhang*, Varsha Kishore*, Felix Wu*, Kilian Q. Weinberger, and Yoav Artzi. Bertscore: Evaluating text generation with bert. In *International Conference on Learning Representations*, 2020. 3, 3

[193] Barret Zoph and Kevin Knight. Multi-source neural translation. *arXiv:1601.00710*, 2016. 2.2, 5.9