Penguins Don’t Fly: Reasoning about Generics through Instantiations and Exceptions

Emily Allaway† Jena D. Hwang‡ Chandra Bhagavatula†
Kathleen McKeown⋆ Doug Downey¶ Yejin Choi†¶
†Columbia University, USA
‡Allen Institute for Artificial Intelligence, USA
¶Northwestern University, USA
†¶Paul G. Allen School of Computer Science & Engineering, University of Washington, USA
eallaway@cs.columbia.edu

Abstract

Generics express generalizations about the world (e.g., birds can fly) that are not universally true (e.g., newborn birds and penguins cannot fly). Commonsense knowledge bases, used extensively in NLP, encode some generic knowledge but rarely enumerate such exceptions and knowing when a generic statement holds or does not hold true is crucial for developing a comprehensive understanding of generics. We present a novel framework informed by linguistic theory to generate EXEMPLARS—specific cases when a generic holds true or false. We generate ∼19k exemplars for ∼650 generics and show that our framework outperforms a strong GPT-3 baseline by 12.8 precision points. Our analysis highlights the importance of linguistic theory-based controllability for generating exemplars, the insufficiency of knowledge bases as a source of exemplars, and the challenges exemplars pose for the task of natural language inference.

1 Introduction

Generics express generalizations (e.g., birds can fly) that allow humans to reason and act with incomplete world knowledge (Asher and Morreau, 1995). Generics allow us to draw plausible inferences about individuals (e.g., Polly is a bird, so Polly can fly) even when we know of counterexamples (e.g., penguins cannot fly). Despite the utility of generalizations, knowledge of counterexamples is necessary for modeling generics and effectively reasoning with them in computational systems.

Recent studies of generics (e.g., Bhagavatula et al., 2022; Bhakthavatsalam et al., 2020) and commonsense KBs (e.g., Speer et al., 2017) provide repositories of generic knowledge. However, these resources rarely mention EXCEPTIONS (i.e., counterexamples) or INSTANTIATIONS (i.e., cases where the generic holds); collectively EXEMPLARS. For systems using these resources as a source of world knowledge, such incomplete information can lead to incorrect deductions (e.g., if Polly is a penguin, which is a bird, then inferring that Polly can fly because birds can fly is false1). Therefore, we propose a novel computational framework that operationalizes linguistic theories in order to automatically generate EXEMPLARS.

In our work, we unify two distinct linguistic theories and use linguistic theory-guided decoding to generate EXEMPLARS. Although large-scale neural language models such as GPT-3 (Brown et al., 2020) have been increasingly successful in few-shot text generation tasks, such generation is both expensive and not easily controllable. Therefore, we instead use the constrained generation algorithm Neurologic A*esque (Lu et al., 2022), which can be applied to any auto-regressive language model; we choose to use GPT-2 (Radford et al., 2019).

1penguin is a bird ∧ birds can fly =⇒ penguins can fly
In this manner, we generate 12562 INSTANTIATIONS and 6297 EXCEPTIONS for ~650 generic statements from Bhagavatula et al. (2022). We conduct human evaluation of the generations and show that our system outperforms few-shot generation by GPT-3 by 12.8 precision points. Our analyses demonstrate not only the importance of linguistic modeling for generating EXEMPLARS and the insufficiency of KBs as a source of EXEMPLARS, but also the challenges EXEMPLARS pose for natural language reasoning.

Our contributions are as follows: (1) we present a novel framework grounded in linguistic theory for representing generics and their EXEMPLARS, (2) we present the first method to automatically generate generic EXEMPLARS and show it outperforms a competitive baseline based on GPT-3 and (3) we present analysis showing the importance of explicit linguistic modeling for this task and the insufficiency of current NLI methods for generics. Our system and data are publicly available2.

2 Related Work

Theory Generics have been studied extensively in semantics, philosophy, and psychology to develop a single logical form for all generics (Lewis and Keenan, 1975; Carlson, 1977, 1989; Krifka, 1987) or a probabilistic definition (Cohen, 1996, 1999, 2004; Kohari et al., 2020), categorize generics (Leslie, 2007, 2008; Khemlani et al., 2009), and analyze specific types (Prasada and Dillingham, 2006, 2009; Haward et al., 2018; Mari et al., 2012; Krifka et al., 2012). Mechanisms to tolerate EXCEPTIONS have also been proposed (Kadmon and Landman, 1993; Greenberg, 2007; Lazaridou-Chatzigoga and Stockall, 2013) but these are primarily theoretical and use carefully chosen examples. In contrast, our work combines these EXCEPTION tolerance mechanisms with generic categorization and proposes a novel, large-scale, computational framework for EXEMPLARS.

Commonsense Knowledge While large-scale CKBs capture a range of commonsense knowledge (Speer et al., 2017; Sap et al., 2019; Forbes et al., 2020; Hwang et al., 2021), they contain necessarily incomplete (i.e., the open-world assumption (Reiter, 1978b)) general knowledge. Furthermore, although recent works have created KBs specifically of generics (Bhakthavatsalam et al., 2020; Bhagavatula et al., 2022) and proposed methods to identify generics in text (Friedrich et al., 2015, 2016), they do not identify or model EXEMPLARS. In our work, we focus directly on automatically generating EXEMPLARS, providing richer commonsense knowledge.

The application of generics to specific individuals is influenced by prototypicality (Rips, 1975; Osherson et al., 1990), with small sets of prototypical norms collected in cognitive science for a range of kinds (Devereux et al., 2014; McRae et al., 2005; Overschelde et al., 2004). However, recent work has shown that neural models have only moderate success at mimicking human prototypicality (Misra et al., 2021; Boratko et al., 2020) or producing commonsense facts without guidance (Petroni et al., 2019). Hence, we combine neural models with a KB of concepts, using linguistic-theory-guided decoding, to generate generics EXEMPLARS.

Reasoning Reasoning with generics is closely related to non-monotonic reasoning (Ginsberg, 1987b,a); specifically default inheritance reasoning (Brewka, 1987; Hanks and McDermott, 1986; Horty and Thomason, 1988; Imielinski, 1985; Poole, 1988; Reiter, 1978a, 1980). Contrary to the proposed solutions for linguistic tests on default inheritance reasoning (Lifschitz, 1989, e.g., can a conclusion about inheritance be inferred based on provided evidence?), later works showed that the presence of generics EXEMPLARS in the evidence impacts what humans perceive as the correct answer (Elio and Pelletier, 1996; Pelletier and Elio, 2005; Pelletier, 2009). These results highlight the importance of identifying generics and accurately modeling their relationships in machine reasoning.

While natural language inference (NLI), a form of deductive reasoning well-studied in NLP (i.a., Dagan et al. (2013); Bowman et al. (2015)), captures notions of inference, studies on non-monotonic reasoning and NLI are limited (Wang et al., 2019; Cooper et al., 1994; Yanaka et al., 2019b,a; Rudinger et al., 2020) and do not include default inheritance reasoning. Therefore, in this work we analyze the interactions between generics EXEMPLARS and NLI and highlight the importance of modeling this relationship in machine reasoning.

3 Framework for EXEMPLARS

A generic statement describes a relation between a concept and a property. Usually, a concept $K$ is a type or kind (e.g., bird) while a property $P$
Table 1: We define three categories of generics with their EXEMPLARS. The logical forms for the generic \((L_G)\) and its EXEMPLARS are also shown below the examples. Using these categories we formulate templates for generating EXEMPLARS (see Table 2). \(K\) is the concept (blue), \(P\) the property (pink). See §3.3 for exoproperty \(\sim P\).

<table>
<thead>
<tr>
<th>Category</th>
<th>Generic ((G))</th>
<th>INSTANTIATION</th>
<th>EXCEPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>(a) quasi-def</td>
<td>“Stars produce radiation”</td>
<td>“The sun produces radiation”</td>
<td>“Stars produce light”</td>
</tr>
<tr>
<td></td>
<td>(K(x) \land r(x, y) \implies \neg P(y))</td>
<td>(K(x) \land r(x, y) \land \neg P(y))</td>
<td>(K(x) \land r(x, y) \land P(y))</td>
</tr>
<tr>
<td>(b) principled</td>
<td>“Birds can fly”</td>
<td>“Owls can fly”</td>
<td>“Penguins can’t fly”</td>
</tr>
<tr>
<td></td>
<td>(K(x) \land P(y))</td>
<td>(r(x, y))</td>
<td>(r(x, y))</td>
</tr>
<tr>
<td>(c) characterizing</td>
<td>“Sharks attack swimmers”</td>
<td>“Threatened sharks attack swimmers”</td>
<td>“Sharks don’t attack swimmers in the shallows”</td>
</tr>
<tr>
<td></td>
<td>(K(x) \land r(x, y) \land \neg P(y))</td>
<td>(K(x) \land r(x, y) \land P(y))</td>
<td>(K(x) \land \neg r(x, y) \land P(y))</td>
</tr>
</tbody>
</table>

No generic has a distinct logical form (see Table 1). For quasi-definitional generics, since the property is defining we assert that the property is logically implied by the concept and relationship together (i.e., \(K \land r \implies P\)). In contrast, for principled generics we assert that the concept and property together logically imply the relationship (i.e., \(K \land P \implies r\)). Finally, for characterizing generics, the logical form depends on whether the generic is interpreted as quasi-definitional or principled. Logical forms with examples are shown in Table 1. Note that in a logical \(L'_G\), the concept \(K\) (property \(P\)) is satisfied by both an individual or subtype of \(K\) (\(P\)).

3.3 Constructing EXEMPLARS

We now define generics EXEMPLARS, deriving them from the logical form of a generic.

### Instantiations

For a generic, INSTANTIATIONS are contextually relevant members of the concept with the desired property. Formally, \(L_G\) (i.e., \(L_G\) with implication replaced by conjunction).

For example, if we have

\[
L_G: \text{BIRD}(x) \land \text{FLY}(y) \implies \text{can}(x, y)
\]

so, \(L'_G: \text{BIRD}(x) \land \text{FLY}(y) \land \text{can}(x, y)\)

then \((x, y)\) = (“owls”, “fly”) satisfies \(L_G\). Note that INSTANTIATIONS all have the same logical form regardless of the generic category (see Table 1).

---

3Description of the theories is provided in §A.2

4For example, \text{BIRD}(x) is true for \(x_1 = “my parrot”\), \(x_2 = “owls”\), and \(x_3 = “these birds”\) since all are birds.

5Replacing the implication with a conjunction excludes instances which satisfy \(L_G\) by satisfying only the right side of the implication (e.g., \text{BIRD}(x) \land \text{FLY}(y) \implies \text{can}(x, y)\) is logically true for \((x, y)\) = (“airplanes”, “fly”) but this is not a valid INSTANTIATION.
**EXCEPTIONS** An EXCEPTION counters one of two interpretations of the generic (see §A.3 for an in-depth discussion). EXCEPTIONS are members of the concept either (i) without the generic property (Greenberg, 2007) or (ii) with an alternative property (i.e., exoproperty) when the generic property is essential to the concept for example. “penguins can’t fly” (for the generic “birds can fly”) counters the interpretation that “all birds can fly”. In contrast, “stars produce light” (for the generic “stars produce radiation”) counters the interpretation that “stars produce only radiation”. Note that in the latter case, there are no specific stars that do not produce radiation. More formally,

**Definition (EXCEPTIONS).** An EXCEPTION satisfies the logical form $\sim L_G$ (i.e., $\sim L_G$ where $\sim P$ replaced by $\sim P$ when applicable). We define $\sim P$ as the exoproperty of $P$: a property adjacent to $P$ that is not $P$ but is contextually relevant to $P$.

For example, if the logical form $L_G$ is

$$\text{STAR}(x) \land \text{produce}(x, y) \implies \text{RADIATION}$$

then $\sim L_G$ is

$$\text{STAR}(x) \land \text{produce}(x, y) \land \sim \text{RADIATION}(y)$$

and so $(x, y)$ (“stars”, “light”) satisfies $\sim L_G$, while $(x, y)$ (“stars”, “movies”) does not. Notice that the latter pair is invalid because “movies” is not a relevant alternative to “radiation” since it is not informative about the generic.

Since a generic’s EXCEPTIONS depend on its logical form $L_G$, they are also dependent on the generic’s category (see Table 1). In particular, the EXCEPTIONS for quasi-definitional generics are individuals with alternative properties (since the property is viewed as essential) while for principled generics the EXCEPTIONS are individuals without the generic property. EXCEPTIONS for characterizing generics are, like the logical forms, dependent on the generic’s interpretation (see §3.2).

### Logical Forms to Templates

Based on our proposed formulae (Table 1) for EXEMPLARS we define seven templates for generation (Table 2). Each template expresses a set of instances that satisfy the logical form of an exemplar. Each template has two sets of content specifications: for the input and for the completion (i.e., the decoder output).

For **instantiations**, we define three templates with subtypes of the concept, property, or both. For **exceptions** we have four templates, using subtypes of either the concept or property (but not both) in order to avoid irrelevant or uninformative instances. For example, for the generic “Birds can fly”, the sentence “Penguins can’t fly long distances” (which has subtypes of both the concept and property) is uninformative because it doesn’t mean penguins can’t fly in general (e.g., they might still be able to fly short distances).

### Methodology

We propose a pipeline system to automatically generate generics EXEMPLARS. Our system takes as input a generic $G$ and the templates derived from its category (§3.4), and outputs a set of generated EXEMPLARS (Fig. 2). First, the system assembles and populates the templates according to the input generic (§4.1, Fig. 2). Then, the filled templates are converted into prompts and constraints that control the generation decoding process (§4.2). Finally, the output is filtered to remove non-viable (§4.3, Fig. 2) or irrelevant (§4.4, Fig. 2) EXEMPLARS.

#### 4.1 Template Assembly

To populate our templates (defined in Table 2), we use a dependency parser to identify the text spans of the concept, relation, and property in a generic. Then, (i) we extract subtypes for the concept and property and use these to construct the (ii) input i.e., generation prompts $x_p$ and (iii) completion (i.e., lexical constraints $C$) specifications.

---

6We assume the generic’s category is known.

7https://spacy.io/
### (i) Subtype Extraction
We extract subtypes using both ConceptNet (Speer et al., 2017)\(^8\) and GPT-3 (Brown et al., 2020). GPT-3 increases the coverage and diversity of subtypes, since many natural and valid subtypes may be missing from ConceptNet (e.g., modifier phrases attached to a concept: “young Arctic fox”). We only use GPT-3 for subtypes of the concept, since by increasing the diversity in the prompt we may encourage diversity in the generated properties (see details Appendix D).

### (ii) Input Specification
We construct the input specifications by constructing generation prompts. Following the template, each prompt consists of either the concept (or a subtype) and the relationship (or its negation) (see Table 2). We prepend to each prompt the generic itself and a connective (e.g., “however”). We rank the prompts by perplexity and use the top \(k_p\) prompts for generation.

### (iii) Completion Specification
Following the templates, we constrain the generation output to describe the property (or a subtype) or its exoproperty (see Table 2). We construct a set of completion constraints (e.g., \(C^{(i)}\) in Fig. 2 specifies “fly” should be in the completion) using lexical items including subtypes, synonyms, and morphological forms.

### 4.2 Generation
In order to generate output that has a specific pragmatic relation to the input without requiring training, we use the NeuroLogic* esque (NeuroLogic*) (Lu et al., 2022) decoding algorithm. NeuroLogic* is an unsupervised decoding algorithm that takes as input a prompt \(x_p\) and set of lexical constraints \(C\) and produces a completion of the prompt \(\hat{y}\) which has high likelihood given the prompt and high satisfaction of the constraints (estimated throughout the decoding). A lexical constraint consists of a set of \(n\)-grams \(w = (w_1^i, \ldots, w_n^p)\) and is satisfied when at least one \(w_i \in w\) is in \(\hat{y}\) (inclusion constraints) or is not in \(\hat{y}\) (exclusion constraints).

By using the input prompts (as \(x_p\)) and completion constraints (as \(C\)) derived from our templates (§4.1), we can control the output content, syntactic form, and pragmatic relevance. We note that since we cannot concretely define the set of relevant potential candidates for a property’s exoproperty (§3.2), decoding constraints must be used to generate EXCEPTIONS.

### Output Ranking
We rank the outputs from NeuroLogic* by prompt and take the top \(k_p\) outputs as potential EXEMPLARS. The outputs are ranked by perplexity (for fluency) and by the probability of a specific NLI label (for relevance) and we average the two ranks. For NLI labels, we hypothesize that a good EXCEPTION aligns with NLI’s contradiction, as does a good INSTANTIATION with entailment (see Fig. 2). While this alignment is useful for ranking, the relationship between the EXEMPLARS and NLI labels is not this straightforward in reality, as we will discuss (§6.3).

### 4.3 Filtering For Viability
Since pre-trained language models have a tendency to hallucinate facts (Rohrbach et al., 2018) or produce non-specific output (e.g., “Birds can do things”), we apply a viability filter to the ranked output generations. Specifically, we train a discriminator to predict whether an output is viable (i.e., true and sufficiently specific that it could be an EXEMPLAR) or not, using human annotated examples (see Appendix B for details). Generations predicted not viable by the trained discriminator are removed from the dataset.

### 4.4 Output Selection
Our final task is to select the generations that are pragmatically relevant (i.e., valid: correctly follows a template) EXEMPLARS. To do this, we first collect gold labels from humans for whether an

---

\(8\) Relations: IsA, InstanceOf, Synonym

---

<table>
<thead>
<tr>
<th>Categories</th>
<th>Template</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>input comp.</td>
<td>Generic Prompt (x_p)</td>
<td>Constraints (C)</td>
</tr>
<tr>
<td>quas-def &amp; char</td>
<td>K + r ≁ (P)</td>
<td>Stars produce radiation “Stars produce” “radiation ∨ x-rays ∨ ...” t1</td>
</tr>
<tr>
<td>&amp; char</td>
<td>K + r ≁ (P)</td>
<td>Birds can fly “Birds can” “flying ∨ ...” t4</td>
</tr>
<tr>
<td>&amp; char</td>
<td>K + r ≁ (P)</td>
<td>Penguins can’t “Penguins can’t” “flying ∨ ...” t5</td>
</tr>
<tr>
<td>&amp; char</td>
<td>K + r ≁ (P)</td>
<td>Birds can fly “Birds can” “flying ∨ ...” t6</td>
</tr>
<tr>
<td>&amp; char</td>
<td>K + r ≁ (P)</td>
<td>Sparrows can’t “Sparrows can’t” “flying ∨ ...” t7</td>
</tr>
<tr>
<td>&amp; char</td>
<td>K + r ≁ (P)</td>
<td>Sun produces radiation “Sun produces” “radiation ∨ x-rays ∨ ...” t2</td>
</tr>
<tr>
<td>&amp; char</td>
<td>K + r ≁ (P)</td>
<td>Birds can fly “Birds can” “flying ∨ ...” t3</td>
</tr>
<tr>
<td>&amp; char</td>
<td>K + r ≁ (P)</td>
<td>Penguins can’t “Penguins can’t” “flying ∨ ...” t4</td>
</tr>
<tr>
<td>&amp; char</td>
<td>K + r ≁ (P)</td>
<td>Birds can fly “Birds can” “flying ∨ ...” t5</td>
</tr>
<tr>
<td>&amp; char</td>
<td>K + r ≁ (P)</td>
<td>Sparrows can’t “Sparrows can’t” “flying ∨ ...” t6</td>
</tr>
<tr>
<td>&amp; char</td>
<td>K + r ≁ (P)</td>
<td>Sun produces radiation “Sun produces” “radiation ∨ x-rays ∨ ...” t2</td>
</tr>
</tbody>
</table>

Table 2: Templates for generating EXEMPLARS, derived from their logical forms (§3.3). \(sub\) indicates a subtype, \(K\) the concept, \(P\) the property, \(\sim P\) its exoproperty (§3.2), \(comp\) is the completion.
EXEMPLARS is valid. These annotations produce two sets of binary labels; one set each for INSTAN-
TIATIONS and EXCEPTIONS. Although this task is more complex than annotating for viability, re-
moving non-viable generations helps reduce the complexity (i.e., we do not need to worry about
false statements that adhere to the template). By an-
notating only viable generations we also reduce the
required amount of annotation. Using the human
annotations, we train two validity discriminators:
one for EXCEPTIONS, one for INSTANTIATIONS.
The trained validity discriminators are used to rank
and select the best generations for each generic as
our system output.

5 Experiment Details
We discuss our experimental setup and specify full
hyperparameters in Appendix C.

5.1 Data Source
We use a subset of the generics dataset from Bhaga-
vatula et al. (2022), a set of 30K generics built upon
common everyday concepts (e.g., “hammers”) and
relations (e.g., “used for”) sourced from resources
such as GenericsKB (Bhakthavatsalam et al., 2020)
and ConceptNet (Speer et al., 2017). The dataset
includes a diverse variety of concepts, including
general knowledge (“Dogs bark”), locative gener-
icics (“In a hotel, you will find a bed”), and com-
parative generics (“Cars are faster than people”).
We use 653 generics from the test set, excluding
human referents as the concept (e.g., nationalities,
professions) due to social bias concerns.

5.2 Annotations
All annotations are done using Amazon Mechanical
Turk with three annotators per HIT (paid at $15/hour on average) and processed using
MACE (Hovy et al., 2013) to filter annotators and
determine the most likely label. We note that while all
tasks achieve moderate inter-annotator agree-
ment, the complex pragmatics of generics make
these tasks difficult for human annotators.

For generic type (§3.1), we conduct two anno-
tation passes to partition all 653 generics into the
groups in Table 1. The Fleiss’ κ (Fleiss, 1971) is
0.41 and 0.58 for the first and second pass re-
spectively. Our categorization results in 296 quasi-
definitional, 125 principled, and 232 characterizing

8Since a generation that is not an INSTANTIATION in not
necessarily an EXCEPTION (and vice versa), these cannot be
directly combined into a single multi-class labeling task.


generics. For the viability filter (§4.3), we an-
notate a set of 7665 system generations from 150
generics. The Fleiss’ κ is 0.53.

For EXEMPLAR gold labels (§4.4), we use separ-
ate annotation tasks for INSTANTIATIONS and EX-
CEPTIONS (see Appendix B for details) with Fleiss’
κ of 0.40 and 0.45 respectively. For training each
discriminator, we randomly sample and annotate
∼1k system generations from ∼300 generics. For
human evaluation (§6.2), we annotate the top 5
discriminator-ranked generations for all generics
from both our system and the baseline.

5.3 Discriminators
For all discriminators, we fine-tune RoBERTa (Liu
et al., 2019). All labeled data is split such that all
generations for a particular generic are in the same
data partition.

5.4 Few-Shot Baseline
As a baseline for generation, we use GPT-3 (Brown
et al., 2020) with few-shot prompting. Since we do
not have access to the decoding algorithm for GPT-
3, we cannot use decoding constraints to control
the output (as in our system). Therefore, we use
few-shot prompting in order to control the output
of GPT-3. Specifically, for each template (Table 2)
we construct a few-shot prompt (Appendix D) that
consists of three examples that illustrate the desired
template. This setup is very similar to the prompts
to our system, except our system is not provided
examples and GPT-3 is not provided with subtypes
(when appropriate to the template). Note, our goal is
not to produce the best possible generations from
GPT-3 but rather to show that constrained genera-
tion from GPT-2 (i.e., NeuroLogic*) outperforms
(and is cheaper and more computationally feasible)
than a natural use of GPT-3.

6 Evaluation
Using our computational framework, we generate
18859 EXEMPLARS for 653 generics (Table 3).

<table>
<thead>
<tr>
<th>Subtype Source</th>
<th>Generated</th>
<th>Output (§4.2)</th>
<th>Viable (§4.3)</th>
<th>Valid (§4.4)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>G3</td>
<td>CN</td>
<td>G3+CN</td>
<td>Excep.</td>
</tr>
<tr>
<td>Total</td>
<td>15358</td>
<td>3501</td>
<td>18859</td>
<td></td>
</tr>
<tr>
<td>Viable</td>
<td>22865</td>
<td>5452</td>
<td>28317</td>
<td></td>
</tr>
<tr>
<td>Exception</td>
<td>42272</td>
<td>10496</td>
<td>52768</td>
<td></td>
</tr>
<tr>
<td>INST.</td>
<td>10983</td>
<td>1579</td>
<td>12562</td>
<td></td>
</tr>
</tbody>
</table>

Table 3: Statistics of the generated dataset, with GPT-3 (G3) and ConceptNet (CN) subtypes used.
Example system generations are in Table 4. To evaluate our approach, we first qualitatively investigate our system and outputs (§6.1) and then conduct a human evaluation (§6.2). We also conduct a detailed analysis of our system and the implications of our results. Our results show that our approach produces a large set of high quality generations for this difficult task. They also highlight current limitations in machine reasoning and potential directions for future work.

### 6.1 Qualitative Analysis

**Observations** We first observe that while close to half the output generations are untrue or not viable, the majority of viable generations are valid EXEMPLARS (Table 3). In addition, we see from system outputs (Table 4) that our system can successfully generate valid EXEMPLARS with subtypes of both the concept (e.g., “angina pectoris” vs. “a chest pain” in (b)) and the property (e.g., “small game” vs. “hunting” in (c)). Furthermore, it produces valid EXCEPTIONS with both the simpler relation-negation templates (i.e., templates t3/t4); see (a) and with relevant exoproperties (i.e., templates t1/t2; see (b) and (c)). These highlight the success of our system in producing high-quality EXEMPLARS.

**Discriminator Analysis** On their respective annotated test sets, the accuracy of the viability discriminator (§4.3) is 75.2 and the accuracies of the trained validity discriminators are 77.4 for INSTANTIATIONS and 75.0 for EXCEPTIONS.

In order to investigate the discriminator quality, we also conduct a manual analysis of the errors made by the validity discriminators. We observe that subtypes are particularly difficult for the discriminators to identify (e.g., that “freshwater lakes and rivers” are a type of “water”). Exoproperties can also be challenging for both the discriminators and humans (e.g., whether “able to land” is a subtype or alternative to “able to move”).

We also observe that the discriminators identify a number of instances that were mislabeled by the human annotators. In particular, for 10 out of the 22 examples (5 out of the 14) where the EXCEPTION (INSTANTIATION) discriminator prediction disagrees with the human label, we judge the discriminator prediction to be correct. For example, “clocks are synchronized to the time zone” is labeled (incorrectly) by humans as an invalid EXCEPTION to the generic “clocks are synchronized to the second”, despite “to the time zone” being a relevant alternative to “to the second”. Counting the human-mislabeled instances as correct would increase the discriminator accuracies to 86% (85%) for the EXCEPTIONS (INSTANTIATIONS).

### 6.2 Human Evaluation

To quantitatively evaluate our system, we compute precision at k (for k = 1 and k = 5) using our human-annotated judgements (§5.2) (Table 5).

Our model outperforms the few-shot baseline (i.e., GPT-3) in all cases, and by a large gap (average 12.8 points). This is especially significant for EXCEPTIONS, which are more challenging to generate than INSTANTIATIONS, and where the baseline performance is close to random. Since generics are defaults, it follows that INSTANTIATIONS should be easier to produce than EXCEPTIONS. The fact that more generated INSTANTIATIONS are true (71% versus 40%) and more true INSTANTIATIONS are accepted by the discriminator (77% versus 50%), compared to the EXCEPTIONS, supports this intuition. Hence, the large improvements by our model over the baseline are significant towards generating these difficult EXCEPTIONS.

Additionally, we examine our model performance across templates. Specifically, we compute the fraction of generations for a template that annotators label as valid, using the same number\(^\text{10}\) of generations for both models for a specific template.

---

\(^{10}\)The models produce similar numbers of generations on all templates except t5.
Table 5: Precision at $k$ ($P@k$).

<table>
<thead>
<tr>
<th>#Gens</th>
<th>$P@1$</th>
<th>$P@5$</th>
<th>$P@1$</th>
<th>$P@5$</th>
</tr>
</thead>
<tbody>
<tr>
<td>GPT-3</td>
<td>0.517</td>
<td>0.563</td>
<td>0.758</td>
<td>0.689</td>
</tr>
<tr>
<td>Ours</td>
<td>0.632</td>
<td>0.616</td>
<td>0.911</td>
<td>0.882</td>
</tr>
</tbody>
</table>

Table 6: Precision by template. #Gens is per template and is the minimum of the models.

(Table 6). We see that not only does our model outperform the baseline for the majority of templates, these templates constitute the majority of the generations (‘#Gens’ in Table 6).

The performance comparison by template does not account for the fact that, while our system is constrained to follow the given template, with GPT-3 the template is only suggested by the prompt and so the model output may not adhere to it. As a result, the GPT-3 performance for certain templates ($t_2$–$t_4$) is inflated because GPT-3 outputs simpler constructions that do not follow the requested template. Therefore, we conduct a manual analysis of the best 40 baseline (i.e., GPT-3) generations per template, ranked by perplexity. For EXCEPTIONS, the baseline produces on average only 2.5–40 generations that fit the desired templates for $t_2$–$t_4$. Additionally, for the one EXCEPTION template, $t_1$, where most baseline generations fit the template (37/40), our model still outperforms the baseline. For INSTANTIATIONS, the baseline performs slightly better (average 10/40 fitting generations) but still poorly. From this we observe that not only is the baseline not controllable, our model outperforms the baseline in cases when it does adhere to output requirements.

6.3 Discussion

Does controllability matter? We ablate the decoding algorithm by removing the constraints (i.e., using beam search) (Table 7a). Although both systems condition their outputs on the same prompts, NeuroLogic*, with linguistic-theory-guided constraints, produces over seven times as many unique generations as unconstrained decoding (i.e., beam search). Additionally, the proportion of valid generations (i.e., accepted by our discriminators) is nearly twice as many for NeuroLogic*. This illustrates the importance of incorporating linguistic-theory-based control into decoding in order to generate a large set of unique, and valid, EXEMPLARS.

Do CKBs contain sufficiently rich information? We probe whether a CKB (i.e., ConceptNet) contains sufficiently rich type information to produce EXEMPLARS. Specifically, we vary the source of subtypes in the template-based prompts and constraints for our system, comparing ConceptNet (CN) to extracting commonsense knowledge from language models (i.e., from GPT-3 prompting (G3) and GPT-2 masked-language model (MLM) (Devlin et al., 2018; Taylor, 1953) infilling).

In fact, CN subtypes result in the fewest generations (Table 7b). In contrast, using GPT-3 for subtypes produces the most generations. Although using MLM for subtypes produces fewer generations than using GPT-3, the proportion of valid generations is comparable and hence MLM could be used as a substitute if using GPT-3 is not feasible. This shows that while CKBs such as ConceptNet are a good source of generics, producing EXEMPLARS requires knowledge that may not always be encoded within the CKB. Therefore, generating EXEMPLARS is important for accessing relevant knowledge beyond what is in CKBs and enabling tools that can effectively use CKBs in reasoning.
NLI label between the generic (premise) and exemplar (hypothesis) and exclude generations that do not have a specific predicted NLI label: ‘contradiction’ for exceptions and ‘entailment’ for instantiations (NLI-sim), ‘neutral’ (NLI-neu), or NLI-sim and ‘neutral’ (NLI-neu-sim). We find that by controlling the NLI relation, we improve precision for exceptions by 20.7 points (Table 8). However, for instantiations NLI label filtering has a negligible impact on precision. Therefore, we observe that controlling NLI relations can improve exception quality but is less beneficial for instantiations. Additionally, note that the alignment with NLI labels is not actually as straightforward as observed, which we discuss next.

**Can NLI sufficiently represent exemplars?**

Although we observe an alignment between predicted NLI labels and exemplars, this actually indicates systematic NLI-model errors, deriving from the insufficiency of NLI schema for capturing the nuances of generics exemplars.

Consider the sentences in Fig. 3, relating to the generic “Birds can fly”. We see that only false statements (i.e., not exceptions) are “unlikely to be true given the information in the premise [generic]” (Dagan et al., 2013) (i.e., NLI contradictions). Since the lack of explicit quantification in generics does not preclude the existence of exceptions, exceptions should actually be labeled neutral by NLI. With instantiations, we observe that the NLI relationship may be either neutral or entailment. These theorized alignments, coupled with our prior observations about exemplars and predicted NLI labels, highlight the challenges of reasoning about exemplars with NLI.

The examples in Fig. 3 also highlight that the NLI neutral label does not distinguish between statements that are true but not entailed or contradictory (e.g., “Penguins cannot fly”) and statements with unknown truth value (e.g., “Tweety bird can fly”). Our generics exemplars emphasize the need for a more fine-grained notion of NLI.

### 7 Conclusion

In this work, we draw on insights from linguistics to propose a novel computational framework to automatically generate valid exemplars for generics, as a step towards capturing the nuances of human reasoning for generics. Our system generates $\sim 19k$ exemplars for 653 generics and outperforms GPT-3 at generating viable examples, while remaining more controllable. We also demonstrate the limitations of CKBs and the importance of explicit linguistic modeling in generating exemplars. That is, the importance of linguistic-theory-based decoding and semantics-based filtering with NLI. Finally, we highlight the inability of current NLI models to reason about and represent the default-inheritance-reasoning relationship between generics and exemplars.

### Acknowledgements

We would like to thank Sarah-Jane Leslie and Maarten Sap for their helpful discussions, the Beaker Team at AI2 for the compute infrastructure, and the anonymous reviewers for their suggestions. This work is supported in part by the National Science Foundation Graduate Research Fellowship under Grant No. DGE-1644869 and by the Office of the Director of National Intelligence (ODNI), Intelligence Advanced Research Projects Activity (IARPA), via the HIATUS Program contract 2022-22072200005. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of ODNI, IARPA, the NSF or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for governmental purposes notwithstanding any copyright annotation therein.

### Limitations and Risks

The generics we source (see §5.1) are exclusively in English. Therefore, our approach may not be suited
to all possible generics in all languages. In particular, our system does not handle generics where valid INSTANTIATIONS include negating (§3.2) the concept. This is due to the restriction that most English generation is left-to-right and it is not possible to define a closed set of possible concept negations for the prompt.

In this work, we do not generate EXEMPLARS for generics involving human referents (e.g., professions, nationalities). We exclude generics involving human referents to mitigate the risk of generating socially biased EXEMPLARS or harmful stereotypes (e.g., “Black folks go to jail for crimes” for the generic “People go to jail for crimes”). Additionally, handling of human stereotypes require methods that are beyond the scope of this paper.

For example, a socially-aware EXCEPTIONS to a generic like “Girls wear dresses” would be “Boys wear dresses, too”. This would require the understanding of the possible subtext of such a statement (e.g. “Only girls wear dresses”), which is beyond the current capabilities of this study and worthy of future exploration.

Finally, we note that while it is not the intended purpose of our system, a malicious user could still use our system to generate EXEMPLARS for a generic involving a person and propagate potentially harmful social biases.

References


Gerhard Brewka. 1987. The logic of inheritance in frame systems. In IJCAI.


Maxwell Forbes, Jena D. Hwang, Vered Shwartz, Maarten Sap, and Yejin Choi. 2020. Social chemistry 101: Learning to reason about social and moral norms. In EMNLP.

Annemarie Friedrich, Alexis Palmer, Melissa Peate Sørensen, and Manfred Pinkal. 2015. Annotating genericity: a survey, a scheme, and a corpus. In LAW@NAACL-HLT.


Matthew L. Ginsberg. 1987b. Readings in nonmonotonic reasoning. In AAAI.


Tomasz Imielski. 1985. Results on translating defaults to circumscription. In IJCAI.


Ximing Lu, Sean Welleck, Peter West, Liwei Jiang, Jungo Kasai, Daniel Khashabi, Ronan Le Bras, Lianhui Qin, Youngjae Yu, Rowan Zellers, Noah A. Smith, and Yejin Choi. 2022. Neurologic a* esque decoding: Constrained text generation with lookahead heuristics. In NAACL.


Hitomi Yanaka, Koji Mineshima, Daisuke Bekki, Kentaro Inui, Satoshi Sekine, Lasha Abzianidze, and Johan Bos. 2019b. *Can Neural Networks Understand Monotonicity Reasoning?* In *BlackboxNLP@ACL*.
A Generics

A.1 Generics and Quantifiers

Explicit quantification (e.g., “Most birds can fly”, “Birds can usually fly”) are excluded from this study because the quantifier implicitly accounts for all potential exceptions. That is, by saying “Most birds can fly” we implicitly indicate that a minority of the birds do not. This being the case, exceptions cannot be generated with statements with explicit quantification.

A.2 Generics Definitions

Categories of Generics  We condense the five generic types proposed by Leslie (2007, 2008) and Khemlani et al. (2009) into our three categories (§3.1). The five types are:

- **Quasi-definitional**: generics concerning properties that are assumed to be universal among a concept. This is the same as our quasi-definitional category, see (a) Table 1. The property is considered a defining characteristic of the concept.

- **L-Principled**: generics concerning properties that are prevalent among a concept and are viewed as inherent, or connected in a principled way (Prasada and Dillingham, 2006, 2009; Haward et al., 2018). These generics are called principled in Leslie (2007, 2008). Note, these generics make up only one half of our “principled” category (§3.1). See first example for category (b) in Table 1; the second example there does not fit Leslie (2007, 2008)’s definition of principled (i.e., L-principled).

- **Striking**: generics describing properties that are uncommon and often dangerous, and members of the concept are disposed to possess them if given the chance (Leslie, 2017). For example, the striking generic “Sharks attack swimmers” assumes all sharks are capable of attacking swimmers. These generics constitute the second half of our “principled” category. See second example (not first) for category (b) in Table 1.

- **Majority characteristic**: generics concerning properties that are neither deeply connected to the concept nor striking but occur in the majority of members of the concept. These constitute one half of our “characterizing category”. See example for (c) in Table 1.

- **Minority characteristic**: generics concerning properties that are neither deeply connected to the concept nor striking but occur in the minority of members of the concept. For example, “Lions have manes”, since only adult male lions (the minority of the lion population) have manes. These constitute the second half of our “characterizing category”.

Both L-principled and striking generics are true in-virtue-of a secondary factor and therefore we group these into one category (i.e., “principled”; see §3.1). For L-principled generics, this may be a factor that causes the property to occur in the concept (e.g., Birds can fly because they have wings). For striking generics, it is the assumed predisposition of the kind to possess the property if given the chance.

For quasi-definitional generics, because the property is considered defining to concept, there is no implied secondary factor in-virtue-of which the generic is true. Therefore, these generics are descriptive and we put them in a separate category from striking and L-principled generics.

Finally, majority and minority characteristic generics are ambiguous in their interpretation. For example, “Lions have manes” can be interpreted as being true in-virtue-of some secondary factor (e.g., as a signal of fitness) or as being a merely accidental relationship. If the interpretation is the former, then lions without manes are valid exceptions (e.g., lion cubs, female lions), while if the interpretation is the latter then other attributes of lions are valid exceptions (e.g., claws, fur).

Focuses of Generics  We note that a generic can focus on the presence of the property within the concept (e.g., “Birds can fly” is concerned with which birds can fly) or can focus on the presence of the concept within holders of the property (e.g., “Triangles have three sides” is more concerned with what concepts have three sides). We will say that the former kind of generic is concept-oriented and the latter is property-oriented. A generic can be both concept and property oriented if it is ambiguous between the two readings (e.g., “Aspirin relieves headaches”).

In this work, we have discussed and used definitions only for concept-oriented generics.

A.3 Exemplars Definitions

Interpretations As discussed (§3.3), exceptions counter an interpretation of the generic.
Importantly, these interpretations must contain universal quantification of either the concept or the property. In this way, the implied universality of the generic can be countered by an exception.

We consider the four interpretations of a generic derived from attaching the universal quantifiers “all” (or “always”) and “only” to either the concept or the property. For example, for the generic “Birds can fly”, we have:

1. All birds can fly.
   ⇒ Concept-oriented:
   which birds (i.e., all) can fly.
2. Only birds can fly.
   ⇒ Property-oriented:
   for being able to fly, how defining (i.e., entirely) are birds.
3. Birds can fly all ways.
   ⇒ Property-oriented:
   for flight, which types (i.e., all of them) can birds do.
4. Birds can only fly.
   ⇒ Concept-oriented:
   of the things birds can do, how defining (i.e., entirely) is flying.

For concept-oriented generics, interpretations 1 and 4 are salient. for property-oriented generics, interpretations 2 and 3 are salient.

The quantifier “only” specifies how defining the concept is for the property (for concept-oriented generics); for property-oriented generics it specifies how the concept is to the property. Hence, quasi-definitional generics (§3.1) correspond to the interpretations containing “only” quantifiers. Therefore, their exceptions will counter the implicit assumption that the property is defining for the concept (or vise versa for property-oriented generics). That is, the exceptions will be members of the concept with other relevant properties. For example, for the generic “Stars produce radiation”, an exception is “The sun produces light”.

On the other hand, the quantifier “all” specifies the prevalence of the property among the concept (for concept-oriented generics). For property-oriented generics, “all” specifies the proportion of the property connected with the concept (e.g., how much of the property can members of the concept do). Hence, the “all” quantifier corresponds to principled generics (§3.1). Note, that even though striking generics (see §A.2) describe very low real-world prevalence, the implication is that prevalence is much higher, since individuals are disposed to possess the property (Leslie, 2017). Therefore, exceptions to principled generics will be members of the concept (or types of the property) that do not possess the desired property (or are not present among the concept). For example, a bird that cannot fly (e.g., a penguin) or a type of movement humans cannot do (e.g., fly, for the generic “humans can move”).

Logical Forms Although we only derived logical forms for concept-oriented generics in this work, similar definitions and logical forms can be derived for property-oriented generics. In particular, only the logical forms for quasi-definitional generics and their exceptions will change if the generic is property-oriented. That is, the $K$ and $P$ in both logicals form for (a) in Table 1 can swapped to obtain the property-oriented versions. In this work, we do not deal with property-oriented generics and their exemplars due to the limitations of English generation (i.e., it is left-to-right).

B Annotation For all annotation tasks, three annotators are used per HIT. When filtering annotators using MACE, we remove annotators with competence below 0.5 (or the median, if lower).

Generic Type Instructions for annotating generic types (§3.1) are shown Figure 4 (for the first pass) and Figure 5 (for the second pass). The first pass categorizes generics as either characterizing or not (either quasi-definitional or principled). The second pass categorizing non-characterizing generics as either quasi-definitional or principled.

Figure 4: Task instructions for first part of the generic type categorization annotation (§5.2).
Viability Task  Instructions for annotating output generations for viability (§4.3) are shown in Figure 6.

True or False? The Task:
- You will be given a sentence. For each sentence:
  - For each sentence, determine whether the sentence is true or false (or indicate that you cannot determine if it is true or false).
  - If the sentence has an adjective or a verb, indicate whether the sentence is true or false.
  - If the sentence is “false” or “true”, indicate why it is true or false.

Exemplars Gold Labels For the INSTANTIATION template generations, annotators are asked whether the generation contradicts the original generic. Instructions are shown in Figure 8. However, for the exception template generations, an exception is not a contradiction of the generic itself but of an associated logical form. For example, “Penguins cannot fly” does not actually contradict the generic itself (“Birds can fly”) but a modified form of the generic involving quantification (i.e., “All birds can fly”). Therefore, we ask annotators whether the generation contradicts two modified forms of the generic. Instructions are shown in Figure 7.

EXEMPLARS Gold Labels For the INSTANTIATION template generations, annotators are asked whether the generation contradicts the original generic. Instructions are shown in Figure 8. However, for the exception template generations, an exception is not a contradiction of the generic itself but of an associated logical form. For example, “Penguins cannot fly” does not actually contradict the generic itself (“Birds can fly”) but a modified form of the generic involving quantification (i.e., “All birds can fly”). Therefore, we ask annotators whether the generation contradicts two modified forms of the generic. Instructions are shown in Figure 7.

EXEMPLARS Gold Labels For the INSTANTIATION template generations, annotators are asked whether the generation contradicts the original generic. Instructions are shown in Figure 8. However, for the exception template generations, an exception is not a contradiction of the generic itself but of an associated logical form. For example, “Penguins cannot fly” does not actually contradict the generic itself (“Birds can fly”) but a modified form of the generic involving quantification (i.e., “All birds can fly”). Therefore, we ask annotators whether the generation contradicts two modified forms of the generic. Instructions are shown in Figure 7.

EXEMPLARS Gold Labels For the INSTANTIATION template generations, annotators are asked whether the generation contradicts the original generic. Instructions are shown in Figure 8. However, for the exception template generations, an exception is not a contradiction of the generic itself but of an associated logical form. For example, “Penguins cannot fly” does not actually contradict the generic itself (“Birds can fly”) but a modified form of the generic involving quantification (i.e., “All birds can fly”). Therefore, we ask annotators whether the generation contradicts two modified forms of the generic. Instructions are shown in Figure 7.

C Implementation Details

C.1 Data We use the generics data from Bhagavatula et al. (2022). For this study, we source from the subset of the test set found to be valid by the discriminator with probability at least 0.5 (768 generics). Of these, we exclude all mentions of human referents (e.g., kinship labels, nationalities, titles, professions) and actions (e.g., studying for a test) to arrive at a dataset of 653 generics. We remove human referents using a seed set of human referent terms compiled based on WordNet (Miller, 1995) and will be provided with the system code. We remove mentions of actions by excluding generics beginning with “In order to”. The dataset is licensed under CC-BY and our usage aligns with the intended use of the data.

Preprocessing We remove adverbs of quantification (i.e., usually, typically, generally) from the generics and exclude generics with verbs of consideration (i.e., consider, posit, suppose, suspect, think). We also convert hedging statements to more explicit forms (e.g., “may have to be” to “must be”).
Table 9: Data split statistics for truthfulness discriminator (§4.3).

<table>
<thead>
<tr>
<th></th>
<th>Train</th>
<th>Dev</th>
<th>Test</th>
<th>All</th>
</tr>
</thead>
<tbody>
<tr>
<td>True</td>
<td>2831</td>
<td>412</td>
<td>433</td>
<td>3676</td>
</tr>
<tr>
<td>False/Non-salient</td>
<td>3180</td>
<td>367</td>
<td>442</td>
<td>3989</td>
</tr>
<tr>
<td>Total</td>
<td>6011</td>
<td>779</td>
<td>875</td>
<td>7665</td>
</tr>
</tbody>
</table>

Table 10: Data split statistics for validity discriminators (§4.4).

<table>
<thead>
<tr>
<th></th>
<th>Train</th>
<th>Dev</th>
<th>Test</th>
<th>All</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Valid</td>
<td>342</td>
<td>35</td>
<td>35</td>
<td>412</td>
</tr>
<tr>
<td>Invalid</td>
<td>462</td>
<td>72</td>
<td>53</td>
<td>587</td>
</tr>
<tr>
<td>Total</td>
<td>804</td>
<td>107</td>
<td>88</td>
<td>999</td>
</tr>
<tr>
<td>INSTANTIATION</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Valid</td>
<td>374</td>
<td>38</td>
<td>29</td>
<td>441</td>
</tr>
<tr>
<td>Invalid</td>
<td>466</td>
<td>38</td>
<td>33</td>
<td>537</td>
</tr>
<tr>
<td>Total</td>
<td>840</td>
<td>76</td>
<td>62</td>
<td>978</td>
</tr>
</tbody>
</table>

**Partitions** The data splits for training the viability discriminator and validity discriminators are shown in Table 9 and Table 10 respectively.

**C.2 Tools**

For extracting components of the generic data we use spacy\(^{11}\) for dependency parsing. We use inflect\(^{12}\) to obtain plural and singular word forms and mlconjug3\(^{13}\) to conjugate verbs. We use nltk\(^{14}\) for additional synonyms.

**C.3 Hyperparameters**

To obtain subtypes from GPT-3 we use the davinci\(^{11}\) model and top-p sampling with \(p = 0.9\), temperature 0.8 and maximum length 100 tokens. We use the top 5 sequences to obtain subtypes. For NLI scores, we use RoBERTa fine-tuned on MNLI (Williams et al., 2018) available from AllenNLP\(^{15}\). For the GPT-3 baseline we use the davinci model and top-p sampling 1.0, temperature 0.8, maximum length 50 tokens and top 5 sequences. Prompts for GPT-3 are given in Appendix D. GPT2-XL has 1.5 billion parameters, GPT-3 has 175 billion parameters. Our experiments are done using Quadro RTX 8000 GPUs.

For generation with NeuroLogic\(^{*}\), we use GPT2-XL (Radford et al., 2019) with a maximum length of 50 tokens and a beam size of 10 with temperature 10000000. We set the constraint satisfaction tolerance to 3. This means that at each step, only candidates whose number of satisfied constraints is within three of the maximum so far are kept. The ‘look ahead’ is also set to 3; look ahead three generation steps during decoding to estimate future constraint satisfaction. During prompt construction, take the top \(k_p = 10\) prompts. If the generic produced less than 10 prompts total, we take half so that low quality prompts are not used even if few are produced. After ranking the output, we keep the top \(k_r = 10\) generations for a template, keeping at most 2 per prompt.

For the viability discriminator, we fine-tune the model for 5 epochs using a batch size of 16 and learning rate \(1e - 5\) and random seed 29725, selected by manual grid search with 27 trials (see bounds Table 11).

For the validity discriminators, we fine-tune the viability discriminator for 3 epochs with a batch size of 16 and learning rate \(3e - 5\). The instantiation discriminator uses a random seed of 4427 and the exception discriminator 4457. Hyperparameters are again selected by manual grid search with 36 trials (see bounds Table 12).

**C.4 Validation Performance**

We show the validation performance for the trained discriminators in Table 13.

**D GPT-3 Prompts**

**D.1 Subtyping**

To obtain subtypes from GPT-3, we first categorize the kinds into six categories: person, animal, other living (e.g., plants), location, temporal (e.g., Thursday), and other (e.g., candle, soup) (Table 14). For

---

\(^{11}\)https://spacy.io/

\(^{12}\)https://pypi.org/project/inflect/

\(^{13}\)https://pypi.org/project/mlconjug3/

\(^{14}\)https://www.nltk.org/

\(^{15}\)https://demo.allennlp.org/textual-entailment/roberta-mnli
Table 13: Trained discriminator accuracy (with mean and variance) on the validation and test sets.

<table>
<thead>
<tr>
<th>Discriminator</th>
<th>Val</th>
<th>Test</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Max</td>
<td>Mean</td>
</tr>
<tr>
<td>Viability</td>
<td>0.771</td>
<td>0.685</td>
</tr>
<tr>
<td>Validity EXCEPTIONS</td>
<td>0.757</td>
<td>0.600</td>
</tr>
<tr>
<td>Validity INSTANTIATIONS</td>
<td>0.763</td>
<td>0.582</td>
</tr>
</tbody>
</table>

each category, we construct a separate prompt for GPT-3 containing one type and five example subtypes. Then, for each kind we use the prompt from its assigned category to obtain subtypes. Note that we exclude all generics where the kind is “person”. This is to avoid producing or repeating stereotypes.

To determine the category, we use seed lists, for person, animal, other living, and locative, or the presence of prepositional beginnings (“On”, “In”, “At”, “During”), for locative and temporal. The “other” category encompasses all kinds that do not fit into another category.

D.2 Few-shot Baseline

The prompts for our few-shot baseline are shown in Table 15. The three examples in the table are provided each on a separate line. Appended to the prompt is a fourth generic and the necessary connective. The same connective is used across all exception (instantiation) templates and is chosen through manual experimentation. We use “But also” for EXCEPTIONS and “For example” for INSTANTIATIONS.
<table>
<thead>
<tr>
<th>Category</th>
<th>Prompt Concept</th>
<th>Prompt Subtypes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Animal</td>
<td>birds</td>
<td>sparrow, canary, large bird, bird of prey, sea bird</td>
</tr>
<tr>
<td>Other living</td>
<td>apple tree</td>
<td>small apple tree, flowering apple tree, apple tree with ripe apples,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>granny smith apple tree, young apple tree</td>
</tr>
<tr>
<td>Locative</td>
<td>hotels</td>
<td>beach hotel, boutique hotel, resort, bed and breakfast, five star hotel</td>
</tr>
<tr>
<td>Temporal</td>
<td>day</td>
<td>morning, hot day, short day, afternoon, evening</td>
</tr>
<tr>
<td>Other</td>
<td>candles</td>
<td>scented candle, advent candle, tealight, candle made from beeswax,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>candle that smells floral</td>
</tr>
<tr>
<td></td>
<td>can of soup</td>
<td>can of tomato soup, can of mushroom bisque, expired can of soup,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>unopened can of soup, organic can of soup</td>
</tr>
</tbody>
</table>

Table 14: Prompts for generating subtypes with GPT-3.

<table>
<thead>
<tr>
<th>Template</th>
<th>Prompt Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>(1)</td>
<td>Elephant are found in zoos. But also elephants are found in the wild in Africa.</td>
</tr>
<tr>
<td></td>
<td>Viruses are spread through body fluids. But also viruses are spread in the air.</td>
</tr>
<tr>
<td></td>
<td>A hair dryer is used to dry hair. But a hair dryer can also be used to dry clothes.</td>
</tr>
<tr>
<td>(2)</td>
<td>Elephant are found in zoos. But also African elephants are found in the wild in Africa.</td>
</tr>
<tr>
<td></td>
<td>Viruses are spread through body fluids. But also coronaviruses are spread in the air.</td>
</tr>
<tr>
<td></td>
<td>A hair dryer is used to dry hair. But also an electric hair dryer can be used to dry clothes.</td>
</tr>
<tr>
<td>(3)</td>
<td>Dogs protect buildings from intruders. But also dogs do not protect apartment buildings from intruders.</td>
</tr>
<tr>
<td></td>
<td>Cowsheds are found on farms. But also cowsheds are not found in orchards.</td>
</tr>
<tr>
<td></td>
<td>The sun produces radiation. But also the sun does not produce x-rays.</td>
</tr>
<tr>
<td>(4)</td>
<td>Birds can fly. For example, seagulls can fly.</td>
</tr>
<tr>
<td></td>
<td>Ducks lay eggs. But also male ducks do not lay eggs.</td>
</tr>
<tr>
<td></td>
<td>Dogs protect buildings from intruders. But also very small dogs do not protect buildings from intruders.</td>
</tr>
<tr>
<td>(5)</td>
<td>Birds can fly. For example, seagulls can fly.</td>
</tr>
<tr>
<td></td>
<td>Dogs protect buildings from intruders. For example, pitbulls protect buildings from intruders.</td>
</tr>
<tr>
<td></td>
<td>Ducks lay eggs. For example, female ducks lay eggs.</td>
</tr>
<tr>
<td>(6)</td>
<td>Viruses are spread through body fluids. For example, viruses are spread through saliva.</td>
</tr>
<tr>
<td></td>
<td>Dogs protect buildings from intruders. For example, dogs protect some private homes from intruders.</td>
</tr>
<tr>
<td></td>
<td>Cowsheds are found on farms. For example, cowsheds are found on dairy farms.</td>
</tr>
<tr>
<td>(7)</td>
<td>Birds can fly. For example, Canadian geese fly long distances to migrate.</td>
</tr>
<tr>
<td></td>
<td>Ostriches lay eggs. For example, female ostriches lay large spotted eggs.</td>
</tr>
<tr>
<td></td>
<td>Elephants are found in zoos. For example, African elephants are found in most large zoos.</td>
</tr>
</tbody>
</table>

Table 15: Prompts for GPT-3 as Few-shot Baseline.