Benchmarking Long-tail Generalization with Likelihood Splits

Ameya Godbole and **Robin Jia**
University of Southern California
{ameyagod,robinjia}@usc.edu

Abstract

In order to reliably process natural language, NLP systems must generalize to the long tail of rare utterances. We propose a method to create challenging benchmarks that require generalizing to the tail of the distribution by re-splitting existing datasets. We create 'Likelihood Splits' where examples that are assigned lower likelihood by a pre-trained language model (LM) are placed in the test set, and more likely examples are in the training set. This simple approach can be customized to construct meaningful train-test splits for a wide range of tasks. Likelihood Splits surface more challenges than random splits: relative error rates of state-of-the-art models increase by 59% for semantic parsing on SPIDER, 93% for natural language inference on SNLI, and 33% for yes/no question answering on BOOLQ, on our splits compared with the corresponding random splits. Moreover, Likelihood Splits create fairer benchmarks than adversarial filtering; when the LM used to create the splits is also employed as the task model, our splits do not unfairly penalize the LM.

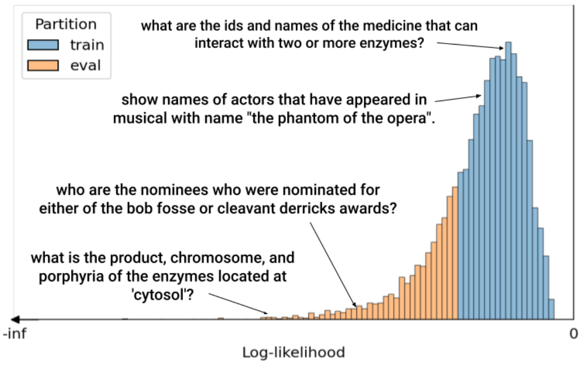

Figure 1: **Likelihood Splits**: We propose to partition the dataset based on likelihood under a language model. The high-likelihood "head" of the distribution becomes the training set while we evaluate generalization to the low-likelihood "tail" of the data. Shown here are queries from the SPIDER dataset in different likelihood buckets: one possible tail generalization could be the handling uncommon entities with known query types.

1 Introduction

Success on in-distribution test data does not necessarily show that a system has solved the underlying task at hand. Systems can achieve artificially high accuracy by exploiting dataset-specific shortcuts, such as spurious feature-label correlations that hold in the data but not in general (Gardner et al., 2021). In many datasets, a large proportion of test examples are similar to training examples, further inflating in-distribution accuracy (Lewis et al., 2021; Czarnowska et al., 2019; Orr et al., 2021). Out-of-distribution (OOD) evaluation paints a clearer picture of a system's ability to perform the task.

Prior work has proposed a variety of methods to test OOD generalization, each with their own strengths and weaknesses. Task-specific behavior tests (Ribeiro et al., 2020; Naik et al., 2018; Gardner et al., 2020) give insights into model behavior but require significant manual (often expert) effort to create. Adversarial data collection, in which annotators try to fool high-performing models (Nie et al., 2020; Potts et al., 2021), also collects challenging examples, but runs the risk of focusing only on a narrow subset of model weaknesses (Bowman and Dahl, 2021; Kaushik et al., 2021). Adversarial filtering removes easy examples from existing datasets (Sakaguchi et al., 2021), but can disproportionately penalize the model used during filtering (Phang et al., 2021). Domain generalization tests transferability to new data domains (Fisch et al., 2019; Miller et al., 2020), but there is no guarantee that generalizing to a given new domain is possible—out-of-domain examples may require skills that are not learnable from the training data (Geiger et al., 2019). Other approaches create dataset splits that test for specific skills, such as length generalization (Lake and Baroni, 2018) and compositional generalization (Shaw et al., 2021), but they only apply to a narrow subset of tasks.

Findings of the Association for Computational Linguistics: EACL 2023, pages 963–983
May 2-6, 2023 ©2023 Association for Computational Linguistics

In this work, we propose **Likelihood Splits**, a general-purpose method to create challenging OOD splits for existing datasets. The principle behind Likelihood Splits is that any system that claims to reliably process natural language must be able to generalize from more common utterances seen during training to the long tail of rare utterances at test time. Generalization, not merely memorization, is necessary because even a very large training dataset cannot exhaustively cover all possible long-tail examples that may be encountered in the real world. Moreover, standard annotation procedures tend to over-sample examples from the head of the distribution, further ignoring the challenge posed by infrequent examples. We identify tail examples using the likelihood under the GPT-2 language model (Radford et al., 2019). Examples with low likelihood under GPT-2 are placed in the held-out evaluation sets and the high likelihood examples are used as the training set (see Figure 1).

Likelihood Splits are a novel, widely applicable strategy that can create interesting generalization benchmarks at no additional annotation cost. They are more challenging than a random split across a wide range of tasks: error rates relative to random splits increase by 59% for T5 (Raffel et al., 2020) on SPIDER (Yu et al., 2018), 93% for ELECTRA (Clark et al., 2020) on SNLI (Bowman et al., 2015), and 33% for ROBERTA (Liu et al., 2019) on BOOLQ (Clark et al., 2019). Moreover, the proposed splits do not unfairly penalize the GPT-2 model used to create the splits when it is used as a task model, thus avoiding one of the downsides of adversarial filtering. We identify many independent challenges required by Likelihood Splits, including generalizing to rare words, complex programs, and syntactically complex sentences. We encourage future benchmark creators to release Likelihood Splits as a complementary evaluation to the standard IID evaluation to better test out-of-distribution generalization performance. We will release the splits discussed in this work along with the code to easily create Likelihood Splits of other datasets.[1]

2 Related Work

Generalizing to the long-tail. Evaluating systems on long-tail phenomena is important, especially because many datasets over-sample the head of the distribution. For example, some question-answering (QA) datasets limit their purview to popular web-pages (Yang et al., 2018) or frequent user queries (Kwiatkowski et al., 2019). Lewis et al. (2021); Liu et al. (2021) demonstrate that models trained on these datasets often fail on examples that do not match the most frequent training cases. Similar observations have been made in entity linking to rare entities, (Orr et al., 2021; Chen et al., 2021), information retrieval for open-domain QA (Sciavolino et al., 2021), relation extraction for rare relations (Sabo et al., 2021) and lexicon induction for rare senses in machine translation (Czarnowska et al., 2019). Zero-shot performance of large LMs on numerical reasoning and factoid questions is also correlated with the frequency of occurence of the facts in the pre-training corpus (Razeghi et al., 2022; Kandpal et al., 2022; Elazar et al., 2022). While we do not test whether models can memorize long-tail knowledge, we instead test whether models can process long-tail sentences. Naik et al. (2022) note that it is challenging to catalogue and evaluate generalization along micro-level dimensions and instead propose benchmarks that vary along macro-level dimensions (such as the language and domain) as a proxy. We hypothesize that LMs learn which micro-level phenomena are rare, as this would improve their overall language modeling objective. In this work, we present a recipe that leverages LMs to evaluate tail generalization for any language task.

Task-specific test sets. Ribeiro et al. (2020) use templated queries to evaluate model performance under various linguistic perturbations. This method requires dataset designers to define phenomena of interest and axes of perturbation along which labels may be preserved or changed. Naik et al. (2018) analyze model errors and instantiate tests that explicitly evaluate models on more examples from each error class. Gardner et al. (2020) check for model consistency under local perturbations of test set examples. All of these approaches require annotators to create new examples, whereas we propose a method to resplit existing datasets.

Adversarial approaches. Søgaard et al. (2021) argue that random splits over-estimate model performance on new in-domain data and recommend the use of adversarial and heuristically challenging splits to estimate generalizability. Adversarial data collection promotes the creation of difficult examples by encouraging annotators to fool a model-in-the-loop (Nie et al., 2020; Potts et al., 2021;

[1] github.com/ameyagodbole/long-tail-likelihood-splits

Kiela et al., 2021). Similarly, Adversarial Filtering removes examples that are easy for a given task model in order to create more challenging benchmarks (Sakaguchi et al., 2021; Yang et al., 2018). However, Kaushik et al. (2021) and Bowman and Dahl (2021) point out that adversarially collected or filtered examples may focus on a narrow set of skills that the "in-the-loop" model lacks, instead of covering all the abilities required for the underlying task. Additionally, the "in-the-loop" task model is disproportionately penalized by the adversarial test sets (Phang et al., 2021). We show in §4.3 that Likelihood Splits do not suffer from this issue.

Domain shift. In NLP, domains can be characterized by the changes in vocabulary and distribution of word use, styles used by authors, and the intended audience. Fisch et al. (2019) pose the challenge of developing QA systems that need to generalize to unseen domains. Miller et al. (2020) show that QA models trained on SQuAD show a performance drop on new domains (while human baseline performance remains unchanged); Miller et al. (2021); Hendrycks et al. (2020) inter alia perform similar analyses of domain shift. SPIDER (Yu et al., 2018) and GRAILQA (Gu et al., 2021) evaluate semantic parsing on unseen table and knowledge base domains respectively. Domain shift is an orthogonal axis of generalization; we focus on generalizing to rare utterances in the same domain.

Out-of-distribution detection. Previous work in OOD detection has used high generative model perplexity as a sign of outliers (Arora et al., 2021; Ren et al., 2019; Lee et al., 2018). Our intuition is similar: low likelihood (high perplexity) is an indicator of rare examples. However, only our work uses likelihood scores for benchmark creation. Moreover, in our setting all examples have been collected under the same data collection protocol, so none of the examples are truly OOD.

Compositional generalization. The ability to "compose" the meaning of a new utterance from the known meaning of its parts (Fodor and Pylyshyn, 1988) is an important aspect of language understanding. The deterministic grammar of programming languages makes semantic parsing, the task of translating a natural language utterance into a logical program, a good testbed for evaluating compositional generalization (Lake and Baroni, 2018; Kim and Linzen, 2020; Hupkes et al., 2020; Keysers et al., 2020; Shaw et al., 2021). However, for tasks where the constituent blocks are not clearly defined, it is unclear how to create such evaluation splits of the data. We compare against compositional generalization splits of the semantic parsing dataset SPIDER (Yu et al., 2018) in §4.

3 Capturing the Tail of the Distribution

In order to find the tail within a dataset, we approximate likelihood of an utterance in the real distribution with its likelihood under a language model (LM). Our method can be easily modified to create meaningful splits for any language task. We demonstrate this by creating Likelihood Splits for:

- SPIDER, a semantic parsing dataset (Yu et al., 2018) consisting of natural language questions and corresponding SQL programs;
- SNLI, a natural language inference dataset (Bowman et al., 2015) consisting of premise and hypothesis sentences paired with labels denoting that the hypothesis is entailed by/neutral to/contradictory to the premise;
- BOOLQ, a question-answering dataset (Clark et al., 2019) consisting of a passages, associated questions, and binary yes/no labels.

3.1 General Approach

We consider language tasks where models must map an input x to an output y (e.g., a SQL query or a label). The input x may be either a single sentence (e.g., semantic parsing) or a pair of sentences (e.g., natural language inference), in which case we write $x = (x_1, x_2)$. Given a dataset D of (x, y) pairs and desired proportion p of evaluation examples, our method partitions D into subsets D_{train} and D_{eval} where $|D_{\text{eval}}| \approx p \cdot |D|$. More specifically, we will first assign a likelihood score $s(x)$ to each $x \in D$, then choose D_{eval} to be the $\lfloor p \cdot |D| \rfloor$ examples in D with lowest value of $s(x)$, and choose $D_{\text{train}} = D \setminus D_{\text{eval}}$. In §3.2, we describe a few different ways to define s. In §3.3, we describe a modification to this procedure that controls for varying length between examples. Finally, we describe task-specific adjustments in §3.4.

3.2 Assigning Likelihood Scores $s(x)$

We use the total log-likelihood over the query tokens assigned by the GPT-2 language model as the score $s(x)$ for every example. There are two ways to use the LM: (1) prompting a frozen LM or (2) fine-tuning the LM on the dataset.

Task	Prompting	Fine-Tuning
SPIDER	`write a database question: {query}`	`<|endoftext|> {query}`
BOOLQ	`Passage: {passage} Ask a question about the passage: {question}`	
SNLI	`Premise: {premise} This hypothesis is {entailed/neutral/a contradiction}: {hypothesis}`	

Table 1: Input formats for single-sentence and sentence-pair tasks in the prompting and fine-tuning settings. Values in curly braces are plugged in from the example. For SNLI, we provide the label in the prompt to prime the LM to the class of hypothesis. The LM is trained (when fine-tuning) and evaluated on generating the query in blue.

Past work has shown that prompting i.e. prepending a task-specific string to the query, helps GPT-2 generalize zero-shot to new tasks (Radford et al., 2019). We use simple prompts that describe the task and prime the LM to the text we expect it to generate (see Table 1). For sentence pair tasks (such as SNLI and BOOLQ), it is necessary to compare the relation between two pieces of text and not just each piece in isolation. Thus, it is intuitive to describe unlikely examples by the conditional likelihood of x_2 given x_1. We demonstrate the flexibility of our approach by providing the label in the prompt if it adds additional information about the text to be generated (e.g. in SNLI).[2] We will refer to this setting which uses the prompted LM with the tag *ll_split pt* in the rest of the work.

The dataset curator may also choose to fine-tune the LM to better capture the task distribution. We fine-tune the GPT-2 LM to maximize either the probability of x for single sentence tasks or the conditional probability of x_2 given the prompt for sentence-pair tasks. When fine-tuning the LM on the dataset, we need to ensure that it is not used to assign scores to the examples it is trained on. Given the dataset D, we first randomly partition D into k folds. For each fold, we fine-tune an LM on the remaining folds and use it to assign log-likelihood scores to examples in the held-out fold. We refer the reader to Appendix A.2 for fine-tuning details. We will refer to this setting as *ll_split* henceforth.

3.3 Controlling for Length

Since the likelihood of an utterance is negatively correlated with its length, we create a split that explicitly controls for the effect of length. After assigning a likelihood score to every utterance, the examples are bucketed based on length (defined by tokenizing the utterance with NLTK (Loper and Bird, 2002)). For single-sentence and sentence-pair tasks, we use the length of the query (x and x_2 respectively) over which log-likelihood was computed. Within each bucket, a fraction p of the examples with the lowest $s(x)$ are put in the evaluation set; aggregating examples from all buckets, $|D_{\text{eval}}| \approx p \cdot |D|$. We will refer to this control setting with the modifier *(-len)* henceforth.[3]

3.4 Dataset-specific Choices and Details

SPIDER. We follow Shaw et al. (2021) and swap examples between the train and evaluation sets such that every logical program atom in the evaluation set appears at least once in the train set. This ensures that the model is not required to generalize to unseen function names and declarations.

SNLI and BOOLQ. We ensure label balance in our splits (as in the original data) by splitting the examples for each label separately, then combining the resulting train and evaluation sets.

Development sets. Csordás et al. (2021) show that without development sets that are in-distribution to challenging test sets, models are prone to over-fitting, which under-estimates their ability to generalize. Thus, after dividing the data into train and evaluation sets, we randomly divide the evaluation set into a development set and test set. Other details are reported in Appendix A.1.

4 Experiments

Next, we benchmark task models on our Likelihood Splits. Splits created using GPT2-medium will be the focus of our analysis. We will briefly study the effect of switching the LM to GPT2-large in §4.4.

When creating Likelihood Splits, the number of folds k for fine-tuning the LM (§3.2) can be chosen by the dataset curator. For results in §4 and §5, we set $k = 3$ arbitrarily. We analyse the effect of

[2] We include the label in the prompt for SNLI but not BOOLQ because the resulting prompts seemed most natural for each dataset. This choice was made before assessing downstream behavior.

[3] We also considered using perplexity, which normalizes for length, but it led to an over-correction where short examples were filtered into the evaluation set.

choosing a different value of k in Appendix A.3. Our results show that the trends and observations discussed here hold true for other values of k.

4.1 Benchmarked Models

One of the goals of this work is to expose long-tail generalization as a challenge to state-of-the-art models; SotA models on the considered benchmarks are all pre-trained models. We make efforts to show that models with different pre-training data and objectives are similarly affected by our proposed splits. Hyperparameters and training details for the reported models are in Appendix A.2.

Semantic parsing. Following Shaw et al. (2021), we benchmark the competitive T5-base model (Raffel et al., 2020) on all splits of the SPIDER dataset. In order to test whether these splits are adversarial to the data splitting language model, we additionally fine-tune GPT2-medium models for the semantic parsing task. To study the effect of model size, we fine-tune T5-small and GPT2-small variants.

SNLI and BOOLQ. We fine-tune two competitive models (ROBERTA (Liu et al., 2019) and ELECTRA (Clark et al., 2020)) at two model sizes (*base* and *large*). Additionally, following Poliak et al. (2018), we train a ROBERTA-large model to perform the task given just the hypothesis. The performance of a hypothesis-only model estimates the degree of spurious correlations that exist in the dataset which give away the label.

4.2 Alternative Splits for Semantic Parsing

We compare the difficulty of the Likelihood Splits with past work on heuristic challenges splits.

Length. Past work has established that text generation models trained on short inputs struggle to generalize to longer inputs at test time (Lake and Baroni, 2018; Hupkes et al., 2020; Newman et al., 2020). We create *Length* splits by placing examples with the longest input queries in the evaluation set and the remaining examples in the training set.

TMCD. Systematicity is the ability to compositionally derive the meaning of an utterance from the known meaning of its parts. Past work studying systematicity in semantic parsing has defined “atoms" as the smallest constituents of the grammar (e.g. variables and function names) and “compounds" as complex structures formed by composing atoms (e.g. multi-argument functions and nested function calls) (Keysers et al., 2020). Following Shaw et al. (2021), we create TMCD (Target Maximum Compound Divergence) splits of SPIDER by maximizing the divergence between the distributions of compounds in the train and evaluation sets.

Split	T5 base	T5 small	GPT-2 medium(Δ)	GPT-2 small
Random	78.6	75.2	69.3 (9.3)	64.7
Length	50.0	44.5	39.9 (10.1)	34.0
Template	60.1	60.0	51.4 (8.7)	45.1
TMCD	66.2	64.1	56.2 (10)	51.4
Split LM: GPT2-medium				
ll_split	66.0	64.2	57.2 (8.8)	51.8
ll_split (-len)	71.3	67.3	59.9 (11.4)	57.3
ll_split pt	60.6	59.7	50.9 (9.7)	45.9
ll_split pt (-len)	73.5	68.4	64.5 (9)	58.3
Split LM: GPT2-large				
ll_split	61.8	61.8	53.7 (8.1)	48.3
ll_split (-len)	69.7	66.2	59.1 (10.6)	54.8
ll_split pt	63.0	58.3	51.4 (11.6)	45.7
ll_split pt (-len)	72.0	70.1	63.4 (8.6)	57.5

Table 2: SPIDER: Exact sequence prediction accuracy for Likelihood Splits created by GPT2-medium and GPT2-large, and other challenge splits. Likelihood Splits are more challenging than random splits while not being adversarial to GPT2-medium. Δ marks the performance drop from T5-base to GPT2-medium.

Template. These splits test the ability of parsers to generate unseen program templates (canonicalized programs formed by anonymizing all variable names and standardizing syntax). We group examples in the SPIDER dataset based on templates defined by Finegan-Dollak et al. (2018). To create the evaluation set, we randomly pick groups of examples till the target set size is reached; the remaining groups form the training set.

4.3 Model Performance on Likelihood Splits

In Table 2, we report exact match accuracy[4] on the data splits using the SPIDER evaluation suite. For SNLI and BOOLQ, we report the accuracy of benchmarked models in Table 3. We create 3 random splits and report mean and standard deviation of accuracy of models trained on each split.

Likelihood Splits are more challenging than random splits. On SPIDER, for example, T5-base accuracy on *ll_split* is 12.6 points lower than the random split accuracy. Likelihood Splits lead to drops in performance that are comparable to the

[4] This metric accounts for the fact that SQL statements are invariant to certain shuffling and change in variable names.

	SNLI					BOOLQ				
	Random	ll_split		ll_split pt		Random	ll_split		ll_split pt	
System			(-len)		(-len)			(-len)		(-len)
RoBERTa-base	89.6 ±0.4	79.3	77.1	82.6	81.7	74.9 ±0.4	71.6	71.2	72.4	71.9
RoBERTa-large	90.5 ±0.5	82.4	79.2	85.0	84.3	84.4 ±0.6	79.3	78.9	82.3	80.6
ELECTRA-base	90.5 ±0.2	80.1	78.4	82.9	82.8	78.8 ±1.1	74.1	74.3	75.2	73.6
ELECTRA-large	91.0 ±1.3	82.6	81.6	85.9	84.9	85.5 ±0.6	82.6	82.1	83.7	81.9
RoBERTa-large (Hypothesis-only)	70.2 ±0.3	64.6	64.6	67.2	69.6	-	-	-	-	-
Human Accuracy	88.7 ±0.8	83.6	84.4	85.2	86.4	-	-	-	-	-

Table 3: SNLI and BOOLQ: Accuracy for various splits and model sizes. Likelihood Splits lead to decreased model performance. Controlling for length further increases the difficulty.

alternative challenge splits. Only Likelihood Splits focus on challenges derived from input language variation; we analyze these challenges in §5.1.

On SNLI and BOOLQ, Likelihood Splits are also more challenging than random splits. For example, ELECTRA-large accuracy decreases by 8.4 points on SNLI and 2.9 points on BOOLQ. On SNLI, the performance of the hypothesis-only baselines on Likelihood Splits is lower than that on the random splits, which indicates that our splits are less easily solved by modeling spurious statistical cues.

Controlling for length preserves challenging nature of splits. Likelihood is negatively correlated with length, so Likelihood Split test data contains longer examples. On SPIDER, generalizing to longer utterances is challenging, so controlling for length makes the Likelihood Splits less challenging. However, these splits are still much more challenging than random splits. For T5-base, *ll_split (-len)* is 7.3 points harder and *ll_split pt (-len)* is 5.1 points harder than the random split. By controlling for length, we identify examples that are more challenging for other reasons (discussed in §5.1). Fitting the dataset distribution with a fine-tuned LM reduces the correlation between length and likelihood on SPIDER. Accordingly, *ll_split pt* poses a stronger length generalization challenge than *ll_split*, and thus is more challenging: T5-base accuracy drops by 18 points on *ll_split pt* compared with the random split.

Conversely, for SNLI and BOOLQ, controlling for length makes the Likelihood Splits slightly harder compared to their uncontrolled versions (ELECTRA-large accuracy drops by 1% from *ll_split* to *ll_split (-len)* on SNLI, and by 0.5% on BOOLQ). This suggests length is not a reason that Likelihood Splits are harder for these datasets. Relatedly, *ll_split pt* is easier than *ll_split* here.

Likelihood Splits do not unfairly penalize the scoring LM. The difference in accuracy between T5-base and GPT2-medium are comparable across all splits (Δ in Table 2). This shows that the Likelihood Splits do not unfairly penalize GPT2-medium, the model used to create the Likelihood Splits. Thus, benchmarks based on Likelihood Splits will be fairer to model class of the LM used.

Human accuracy is less affected. We estimate human accuracy on the evaluation sets using the ∼10% of examples that were annotated with 5 labels in the original SNLI dataset. Human accuracy is at most 5.1% lower on our proposed splits than on the random splits. Model performance drops more severely than the smaller drop in human accuracy; models that were previously superhuman are now worse than the estimated human performance (except for ELECTRA-large on *ll_split pt*). In comparison, adversarial filtering (Le Bras et al., 2020) has a larger drop in human accuracy from 88% on the standard split to 78% on their most challenging split. Thus, our method does not as heavily emphasize mislabeled or ambiguous examples.

4.4 Effect of the LM on Likelihood Splits.

We study the effect changing the language model by using a GPT2-large model to create the Likelihood Splits of SPIDER. The log-likelihood scores assigned to the examples by GPT2-medium and GPT2-large are highly correlated; Pearson correlation coefficient (r) between log-likelihood scores from fine-tuned models is 0.96 while it is 0.99 for the pre-trained models. Accounting for swapping of examples in order to meet the atom constraint, the evaluation sets differ in 16% of examples in the *ll_split* setting, and 10% of the examples in *ll_split pt* setting. *ll_split* is more challenging when using GPT2-large; T5-base accuracy drops an additional

System	Random	ll_split		ll_split reverse	
			(-len)		(-len)
		SPIDER			
T5-base	78.6	66.0	71.3	83.9	81.5
		SNLI			
E-base	90.5 ±0.2	80.1	78.4	96.6	97.4
E-large	91.0 ±1.3	82.6	81.6	96.7	97.4
		BOOLQ			
E-base	78.8 ±1.1	74.1	74.3	77.8	78.1
E-large	85.5 ±0.6	82.6	82.1	85.8	86.8

Table 4: Accuracy on SPIDER, SNLI and BOOLQ when training on the unlikely (tail) queries and evaluating on the likely (head) queries (*ll_split reverse*). The model accuracy on the reverse splits are comparable to or higher than accuracy on the random split. This supports the claim that generalizing to rare instances is a significant challenge. (E-base and E-large are ELECTRA models)

4.2% compared to the *ll_split* with GPT2-medium. The other splits are comparable with accuracies differing by 1-2% across all models (see Table 2). Thus, we expect splits created with different LMs to demonstrate similar characteristics.

4.5 Are Reverse Likelihood Splits Difficult?

We wish to test whether the decrease in task accuracy is driven by rarity of the instances or whether any likelihood based distribution shift is challenging. We test this hypothesis by creating a setting that requires generalizing from the tail of the distribution to the head. Using the same likelihoods as before, we create reverse splits where the more likely (head) of the distribution is used as the evaluation set instead of the unlikely (tail). From Table 4, we see that the accuracy of ELECTRA-large on SNLI increases from 81.6% on the Likelihood Split to 96.7% on the reversed split. For comparison, this is more than the accuracy on random splits of SNLI (91%). We see similar trends on BOOLQ and SPIDER where the reverse splits are as easy as or easier than the corresponding random splits. We conclude that generalizing specifically to the tail is what makes our splits difficult.

5 Analysis of Data Splits

In order to highlight the challenges posed by our proposed splits, we analyze how the development sets (to ensure unseen test sets) differ from the training sets in each split. Our splits require models to simultaneously excel at many different skills believed to be important for language understanding.

5.1 Properties of the Proposed SPIDER Splits

TMCD-related properties and length. Following Shaw et al. (2021), we report atom and compound divergences of the various splits in Table 13 of Appendix A.4. Divergence measures how much the distribution of atoms/compounds differs between the train and evaluation set. Our approach leads to splits with higher than random atom divergence, which shows that our split poses the challenge of **generalizing to rare atoms**. Similarly, a greater than random compound divergence emerges from the resulting split. This means that the split also requires some amount of **compositional generalization**. From Figure 6 in Appendix A.6, we see that log-likelihood preferentially puts the longer queries in the test set and the corresponding length variation is closer to that of the length split than the other splits. Hence, it naturally requires some aspect of **length generalization**. As expected, by controlling for length of the utterances, we can remove the challenge of length generalization.

Program difficulty. SPIDER assigns a rating of 'easy', 'medium', 'hard', or 'extra hard' to every SQL program. From Figure 2, we see that the evaluation sets of the Likelihood Splits contain more examples from the harder categories than the training sets. Controlling for length reduces this effect but does not completely remove it (see Appendix A.5 for more details). Note that this skew emerges even though we do not consider the programs when creating these splits.

Rare words. On the input side, we first analyze the distribution of rare words. We define rare words as all English words[5] in the SPIDER dataset that occur at most 1 time per million words according to SUBTLEXus (Brysbaert and New, 2009). This results in a list of 561 words. We report the fraction of words in the development set that are rare. This metric automatically controls for the length of the examples; longer examples are more likely to contain rare words by chance. To estimate the distribution of this fraction under random splits (null distribution), we create 500 random splits and plot the distribution of values observed. From Figure 3, we observe that the Likelihood Splits have more rare words in the test set, especially for the *ll_split*

[5] We filter out incorrect spellings using the word list at https://github.com/dwyl/english-words

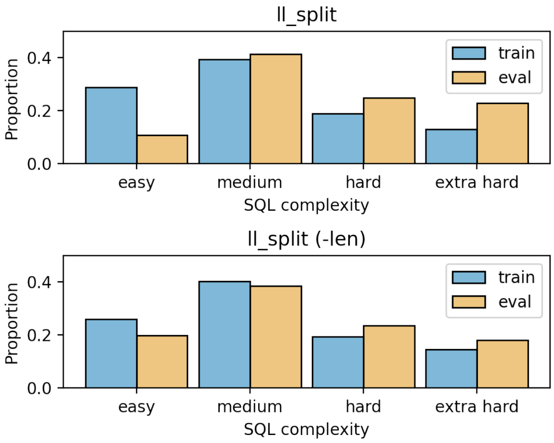

Figure 2: SPIDER: Distribution of SQL programs of varying complexity in the train and development set of Likelihood Splits. These splits show a skew towards training on easy examples and evaluating on harder examples.

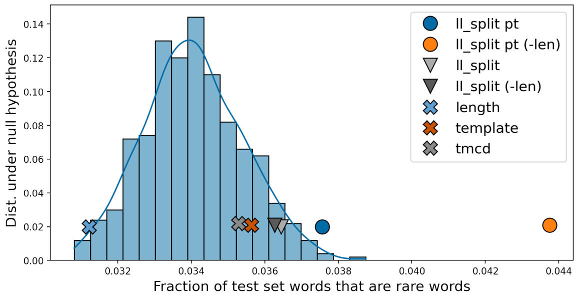

Figure 3: SPIDER: Statistics of the fraction of dev set words that are rare for various splits. This is plotted against the distribution of values observed for 500 random slits of the data. ll_split variants retain a larger fraction of rare words in the test set. Controlling for length finds shorter examples with more rare words.

Category	Random	*ll_split pt*
Easy	92.3% (.225)	81.3% (.077)
Medium	82.3% (.409)	71.6% (.435)
Hard	78.4% (.201)	60.1% (.233)
Extra Hard	62.9% (.164)	52.5% (.254)
Dev set Acc	80.6%	64.8%
Projected Acc		77.15%

Table 5: SPIDER: Accuracy of T5-base aggregated by the SQL hardness rating for random and *ll_split pt* dev sets. The number in brackets is the fraction of dev set examples that fall in each bucket. The examples in the dev set of *ll_split pt* are skewed towards harder examples. However, performance of T5-base on *ll_split pt* is lower than performance on random split in every bucket. Projecting and re-weighting the random set accuracies using the fraction of examples in each bucket in *ll_split pt* over-estimates dev set performance.

pt setting. Controlling for length puts shorter examples in the evaluation sets, but a larger fraction of the words are rare. The other challenging splits considered do not focus on the input language variation and hence the fraction of development set words that are rare is closer to random.

Input syntactic complexity. We also study the query parse tree structures in various splits of the dataset in Figure 8. We measure the complexity of the parse tree based on mean and max depth as well as Yngve score (Yngve, 1960) which is a measure of syntactic complexity. We see that more complex queries tend to be assigned lower likelihood and correspondingly put in the evaluation set. The effect of the complexity is also correlated with length and balancing for length reduces the gap between the complexity of the train and test set. We refer the reader to Appendix A.7 for more details.

Effect on accuracy. In Appendix A.8 and Table 5, we show that the higher frequency of both novel compounds (i.e., compounds not seen during training) and harder programs each partially explain the higher difficulty of *ll_split pt* for T5-base. For example, 16% of dev examples in the random split have 'extra hard' programs, compared with 25% in *ll_split pt*. On the random split, T5-base gets 63% of these examples correct, compared with 81% dev accuracy overall, so these examples are indeed more challenging. On 'extra hard' examples in *ll_split pt*, T5-base has an even lower accuracy of 53%. Thus, the mere fact that *ll_split pt* has more 'extra hard' examples does not fully explain why it is harder; other factors must also be playing a role.

5.2 Properties of the Proposed SNLI Splits

For SNLI, we study the variation of premise and hypothesis length (A.9), distribution of rare words (A.10), Yngve score (Yngve, 1960) for syntactic complexity (A.11), and Flesch-Kincaid (Kincaid et al., 1975) reading grade-level (A.12). Evaluation sets of Likelihood Splits of SNLI are more complex than their corresponding training sets on all 4 variations; evaluation set examples tend to be longer, tend to contain more rare words, are more syntactically complex, and have higher reading levels.

Controlling for length removes length variation, and slightly decreases the skew in reading level. Surprisingly, the Yngve scores of evaluation examples are skewed to being less complex than the

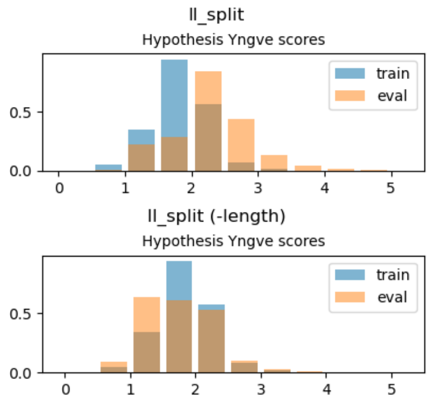

Figure 4: SNLI: Distribution of Yngve scores computed on the parse tree of the hypothesis. The evaluation sets for *ll_split* contain more complex utterances. Normalizing for the length surprisingly reverses the skew.

corresponding training set (see Figure 4), even though the length controlled variants of SNLI are more challenging than the corresponding Likelihood Splits. Some the difficulty when controlling for length can be explained by the increased proportion of rare words.

We analyze the errors of RoBERTa-large on the development set of *ll_split (-len)*, the hardest SNLI split (see Appendix A.13 for concrete examples). We find several instances of examples that require common-sense or world knowledge to be solved correctly. These include knowledge of terms such as crowd-surfing and lincoln logs (a type of toy), and facts like zip-lining is an exciting activity. We find that a small fraction of the errors are caused by ambiguous or incorrect labels. There are a several instances of spelling mistakes, a few of which change the meaning of the sentence.

6 Conclusion

With the saturation of static, single-metric leaderboards, there is growing consensus for the development of holistic evaluation benchmarks. This includes evaluation of systems on aspects of performance beyond just single error rate on in-distribution data; aspects such as performance on out-of-distribution data (Linzen, 2020), and evaluating generalizability, robustness and fairness (Ethayarajh and Jurafsky, 2020). In this work, we describe an approach to benchmark long-tail generalization, a necessary skill for NLP systems that truly understand language. We demonstrate the challenge posed by our splits to state-of-the-art models on several tasks; standard evaluation overestimates model performance on long-tail utterances. Instead of releasing a random split as the only metric on official benchmarks, our simple method can be used, for a wide range of tasks, to expose additional challenges in the collected data at no annotation cost. Benchmarking long-tail generalization in this manner, can test model behavior on a broad set of generalization challenges, which may be missed by evaluations that test specific skills in isolation.

Limitations

Evaluating a proposed benchmarking method such as ours is challenging, as there is no community consensus on what properties characterize an ideal benchmark. While we have argued that Likelihood Splits have a number of desirable properties, ultimately we intend Likelihood Splits to *complement* other options for creating benchmarks, not replace them. In particular, we do not aim to replace methods that require additional annotation and domain knowledge discussed in §2. In situations where previously collected datasets contain no or very few examples of a particular type, creating new data may be the only way to test models on that type of example. We view our approach as one lightweight option that dataset curators can choose to create a more holistic benchmark.

The properties of the Likelihood Splits that we have studied in this work do not fully explain what makes the Likelihood Splits harder. Dataset splits that explicitly test specific skills like length generalization and compositional generalization are good at exposing specific weaknesses in models. While it is hard to pinpoint the source of difficulty, our approach is complementary in that it can test a much broader set of skills that a narrow test may miss.

The difficulty of out-of-distribution generalization is higher in low resource languages, however, we show that the problem is yet not solved for NLP tasks even in the high resource English language. Our approach has the flexibility to use any autoregressive LM to score the utterances; large multi-lingual LMs such as BLOOM (Scao et al., 2022) can be used if appropriate.

The model performance gaps between random split and Likelihood Splits are small (2-4%) on some datasets (e.g. BOOLQ). We cannot guarantee that Likelihood Splits for a new dataset will be much more challenging than random splits. In such a situation, other complementary evaluation strategies may be recommended to more strenuously challenge models.

Our approach has multiple knobs to control the properties of the splits created: (1) prompting/finetuning the LM, (2) controlling length variation, and (3) dataset specific choices such as label balancing. This choice gives dataset curators a lot of control to modify the approach. It is possible that the behaviour of the splits might be inconsistent under some changes. In our experiments, we find that qualitative findings are largely consistent, even across changes such as using a different language model.

Finally, there is no guarantee that the challenge posed is a fair generalization task (Geiger et al., 2019); we cannot guarantee that all skills needed to solve the test set can be learned from the training set. Nevertheless, since our approach partitions data that was collected under a single consistent protocol, it is more likely to be fair than methods that rely on an additional, separate annotation process to create test data.

Acknowledgements

We thank Xiang Ren for discussions and feedback in the initial stages of the work. We thank Rajarshi Das and colleagues at USC for feedback on drafts of this work. This work was funded in part by a gift from Open Philanthropy.

References

Udit Arora, William Huang, and He He. 2021. Types of out-of-distribution texts and how to detect them. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 10687–10701, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 632–642, Lisbon, Portugal. Association for Computational Linguistics.

Samuel R. Bowman and George Dahl. 2021. What will it take to fix benchmarking in natural language understanding? In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4843–4855, Online. Association for Computational Linguistics.

Marc Brysbaert and Boris New. 2009. Moving beyond Kucera and Francis: a critical evaluation of current word frequency norms and the introduction of a new and improved word frequency measure for American English. *BEHAVIOR RESEARCH METHODS*, 41(4):977–990.

Anthony Chen, Pallavi Gudipati, Shayne Longpre, Xiao Ling, and Sameer Singh. 2021. Evaluating entity disambiguation and the role of popularity in retrieval-based NLP. 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 4472–4485, Online. Association for Computational Linguistics.

J.K Chung, P.L Kannappan, C.T Ng, and P.K Sahoo. 1989. Measures of distance between probability distributions. *Journal of Mathematical Analysis and Applications*, 138(1):280–292.

Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. 2019. BoolQ: Exploring the surprising difficulty of natural yes/no questions. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2924–2936, Minneapolis, Minnesota. Association for Computational Linguistics.

Kevin Clark, Minh-Thang Luong, Quoc V. Le, and Christopher D. Manning. 2020. ELECTRA: Pre-training text encoders as discriminators rather than generators. In *ICLR*.

Róbert Csordás, Kazuki Irie, and Jürgen Schmidhuber. 2021. The devil is in the detail: Simple tricks improve systematic generalization of transformers.

Paula Czarnowska, Sebastian Ruder, Edouard Grave, Ryan Cotterell, and Ann Copestake. 2019. Don't forget the long tail! a comprehensive analysis of morphological generalization in bilingual lexicon induction. 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 974–983, Hong Kong, China. Association for Computational Linguistics.

Yanai Elazar, Nora Kassner, Shauli Ravfogel, Amir Feder, Abhilasha Ravichander, Marius Mosbach, Yonatan Belinkov, Hinrich Schütze, and Yoav Goldberg. 2022. Measuring causal effects of data statistics on language model's 'factual' predictions.

Kawin Ethayarajh and Dan Jurafsky. 2020. Utility is in the eye of the user: A critique of NLP leaderboards. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4846–4853, Online. Association for Computational Linguistics.

Catherine Finegan-Dollak, Jonathan K. Kummerfeld, Li Zhang, Karthik Ramanathan, Sesh Sadasivam, Rui

Zhang, and Dragomir Radev. 2018. Improving text-to-SQL evaluation methodology. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 351–360, Melbourne, Australia. Association for Computational Linguistics.

Adam Fisch, Alon Talmor, Robin Jia, Minjoon Seo, Eunsol Choi, and Danqi Chen. 2019. MRQA 2019 shared task: Evaluating generalization in reading comprehension. In *Proceedings of 2nd Machine Reading for Reading Comprehension (MRQA) Workshop at EMNLP*.

J. Fodor and Z. Pylyshyn. 1988. Connectionism and cognitive architecture: A critical analysis. *Cognition*, 28:3–71.

Matt Gardner, Yoav Artzi, Victoria Basmov, Jonathan Berant, Ben Bogin, Sihao Chen, Pradeep Dasigi, Dheeru Dua, Yanai Elazar, Ananth Gottumukkala, Nitish Gupta, Hannaneh Hajishirzi, Gabriel Ilharco, Daniel Khashabi, Kevin Lin, Jiangming Liu, Nelson F. Liu, Phoebe Mulcaire, Qiang Ning, Sameer Singh, Noah A. Smith, Sanjay Subramanian, Reut Tsarfaty, Eric Wallace, Ally Zhang, and Ben Zhou. 2020. Evaluating models' local decision boundaries via contrast sets. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 1307–1323, Online. Association for Computational Linguistics.

Matt Gardner, William Merrill, Jesse Dodge, Matthew Peters, Alexis Ross, Sameer Singh, and Noah A. Smith. 2021. Competency problems: On finding and removing artifacts in language data. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 1801–1813, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Atticus Geiger, Ignacio Cases, Lauri Karttunen, and Christopher Potts. 2019. Posing fair generalization tasks for natural language inference. 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 4485–4495, Hong Kong, China. Association for Computational Linguistics.

Yu Gu, Sue Kase, Michelle Vanni, Brian Sadler, Percy Liang, Xifeng Yan, and Yu Su. 2021. Beyond i.i.d.: Three levels of generalization for question answering on knowledge bases. In *Proceedings of the Web Conference 2021*, WWW '21, page 3477–3488, New York, NY, USA. Association for Computing Machinery.

Dan Hendrycks, Xiaoyuan Liu, Eric Wallace, Adam Dziedzic, Rishabh Krishnan, and Dawn Song. 2020. Pretrained transformers improve out-of-distribution robustness. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2744–2751, Online. Association for Computational Linguistics.

Dieuwke Hupkes, Verna Dankers, Mathijs Mul, and Elia Bruni. 2020. Compositionality decomposed: How do neural networks generalise? *Journal of Artificial Intelligence Research*, 67:757–795.

Nikhil Kandpal, Haikang Deng, Adam Roberts, Eric Wallace, and Colin Raffel. 2022. Large language models struggle to learn long-tail knowledge.

Divyansh Kaushik, Douwe Kiela, Zachary C. Lipton, and Wen-tau Yih. 2021. On the efficacy of adversarial data collection for question answering: Results from a large-scale randomized study. 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 6618–6633, Online. Association for Computational Linguistics.

Daniel Keysers, Nathanael Sch"arli, Nathan Scales, Hylke Buisman, Daniel Furrer, Sergii Kashubin, Nikola Momchev, Danila Sinopalnikov, Lukasz Stafiniak, Tibor Tihon, Dmitry Tsarkov, Xiao Wang, Marc van Zee, and Olivier Bousquet. 2020. Measuring compositional generalization: A comprehensive method on realistic data. In *ICLR*. Additional citation for MCD splits.

Douwe Kiela, Max Bartolo, Yixin Nie, Divyansh Kaushik, Atticus Geiger, Zhengxuan Wu, Bertie Vidgen, Grusha Prasad, Amanpreet Singh, Pratik Ringshia, Zhiyi Ma, Tristan Thrush, Sebastian Riedel, Zeerak Waseem, Pontus Stenetorp, Robin Jia, Mohit Bansal, Christopher Potts, and Adina Williams. 2021. Dynabench: Rethinking benchmarking in NLP. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4110–4124, Online. Association for Computational Linguistics.

Najoung Kim and Tal Linzen. 2020. COGS: A compositional generalization challenge based on semantic interpretation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 9087–9105, Online. Association for Computational Linguistics.

J. Peter Kincaid, Robert P. Fishburne, R L Rogers, and Brad S. Chissom. 1975. Derivation of new readability formulas (automated readability index, fog count and flesch reading ease formula) for navy enlisted personnel. Institute for Simulation and Training.

Nikita Kitaev and Dan Klein. 2018. Constituency parsing with a self-attentive encoder. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2676–2686, Melbourne, Australia. Association for Computational Linguistics.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Matthew Kelcey, Jacob Devlin, Kenton Lee, Kristina N. Toutanova, Llion Jones, Ming-Wei Chang, Andrew Dai, Jakob Uszkoreit, Quoc Le, and Slav Petrov. 2019. Natural questions: a benchmark for question answering research. *Transactions of the Association of Computational Linguistics*.

Brenden M. Lake and Marco Baroni. 2018. Generalization without systematicity: On the compositional skills of sequence-to-sequence recurrent networks. In *ICML*.

Ronan Le Bras, Swabha Swayamdipta, Chandra Bhagavatula, Rowan Zellers, Matthew E. Peters, Ashish Sabharwal, and Yejin Choi. 2020. Adversarial filters of dataset biases. In *ICML*.

Kimin Lee, Kibok Lee, Honglak Lee, and Jinwoo Shin. 2018. A simple unified framework for detecting out-of-distribution samples and adversarial attacks. In *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc.

Patrick Lewis, Pontus Stenetorp, and Sebastian Riedel. 2021. Question and answer test-train overlap in open-domain question answering datasets. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 1000–1008, Online. Association for Computational Linguistics.

Tal Linzen. 2020. How can we accelerate progress towards human-like linguistic generalization? In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5210–5217, Online. Association for Computational Linguistics.

Linqing Liu, Patrick S. H. Lewis, Sebastian Riedel, and Pontus Stenetorp. 2021. Challenges in generalization in open domain question answering. *CoRR*, abs/2109.01156.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Edward Loper and Steven Bird. 2002. Nltk: The natural language toolkit. In *Proceedings of the ACL-02 Workshop on Effective Tools and Methodologies for Teaching Natural Language Processing and Computational Linguistics - Volume 1*, ETMTNLP '02, page 63–70, USA. Association for Computational Linguistics.

John Miller, Karl Krauth, Benjamin Recht, and Ludwig Schmidt. 2020. The effect of natural distribution shift on question answering models. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 6905–6916. PMLR.

John P Miller, Rohan Taori, Aditi Raghunathan, Shiori Sagawa, Pang Wei Koh, Vaishaal Shankar, Percy Liang, Yair Carmon, and Ludwig Schmidt. 2021. Accuracy on the line: on the strong correlation between out-of-distribution and in-distribution generalization. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 7721–7735. PMLR.

Aakanksha Naik, Jill Lehman, and Carolyn Rosé. 2022. Adapting to the long tail: A meta-analysis of transfer learning research for language understanding tasks. *Transactions of the Association for Computational Linguistics*, 10:956–980.

Aakanksha Naik, Abhilasha Ravichander, Norman Sadeh, Carolyn Rose, and Graham Neubig. 2018. Stress test evaluation for natural language inference. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 2340–2353, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

Benjamin Newman, John Hewitt, Percy Liang, and Christopher D. Manning. 2020. The EOS decision and length extrapolation. In *Proceedings of the Third BlackboxNLP Workshop on Analyzing and Interpreting Neural Networks for NLP*, pages 276–291, Online. Association for Computational Linguistics.

Yixin Nie, Adina Williams, Emily Dinan, Mohit Bansal, Jason Weston, and Douwe Kiela. 2020. Adversarial NLI: A new benchmark for natural language understanding. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4885–4901, Online. Association for Computational Linguistics.

Laurel J. Orr, Megan Leszczynski, Neel Guha, Sen Wu, Simran Arora, Xiao Ling, and Christopher Ré. 2021. Bootleg: Chasing the tail with self-supervised named entity disambiguation. In *11th Conference on Innovative Data Systems Research, CIDR 2021, Virtual Event, January 11-15, 2021, Online Proceedings*. www.cidrdb.org.

Jason Phang, Angelica Chen, William Huang, and Samuel R. Bowman. 2021. Adversarially constructed evaluation sets are more challenging, but may not be fair.

Adam Poliak, Jason Naradowsky, Aparajita Haldar, Rachel Rudinger, and Benjamin Van Durme. 2018. Hypothesis only baselines in natural language inference. In *Proceedings of the Seventh Joint Conference on Lexical and Computational Semantics*, pages 180–191, New Orleans, Louisiana. Association for Computational Linguistics.

Christopher Potts, Zhengxuan Wu, Atticus Geiger, and Douwe Kiela. 2021. DynaSent: A dynamic benchmark for sentiment analysis. 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 2388–2404, Online. Association for Computational Linguistics.

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

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

Yasaman Razeghi, Robert L. Logan, Matt Gardner, and Sameer Singh. 2022. Impact of pretraining term frequencies on few-shot reasoning.

Jie Ren, Peter J. Liu, Emily Fertig, Jasper Snoek, Ryan Poplin, Mark Depristo, Joshua Dillon, and Balaji Lakshminarayanan. 2019. Likelihood ratios for out-of-distribution detection. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.

Marco Tulio Ribeiro, Tongshuang Wu, Carlos Guestrin, and Sameer Singh. 2020. Beyond accuracy: Behavioral testing of NLP models with CheckList. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4902–4912, Online. Association for Computational Linguistics.

Brian Roark, Margaret Mitchell, and Kristy Hollingshead. 2007. Syntactic complexity measures for detecting mild cognitive impairment. In *Biological, translational, and clinical language processing*, pages 1–8, Prague, Czech Republic. Association for Computational Linguistics.

Ofer Sabo, Yanai Elazar, Yoav Goldberg, and Ido Dagan. 2021. Revisiting few-shot relation classification: Evaluation data and classification schemes. *Transactions of the Association for Computational Linguistics*, 9:691–706.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. 2021. Winogrande: An adversarial winograd schema challenge at scale. *Commun. ACM*, 64(9):99–106.

Teven Le Scao, Angela Fan, Christopher Akiki, Ellie Pavlick, Suzana Ilić, Daniel Hesslow, Roman Castagné, Alexandra Sasha Luccioni, François Yvon, Matthias Gallé, et al. 2022. Bloom: A 176b-parameter open-access multilingual language model.

Christopher Sciavolino, Zexuan Zhong, Jinhyuk Lee, and Danqi Chen. 2021. Simple entity-centric questions challenge dense retrievers. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 6138–6148, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Peter Shaw, Ming-Wei Chang, Panupong Pasupat, and Kristina Toutanova. 2021. Compositional generalization and natural language variation: Can a semantic parsing approach handle both? 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 922–938, Online. Association for Computational Linguistics.

Anders Søgaard, Sebastian Ebert, Jasmijn Bastings, and Katja Filippova. 2021. We need to talk about random splits. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 1823–1832, Online. Association for Computational Linguistics.

Eric Wallace, Adina Williams, Robin Jia, and Douwe Kiela. 2021. Analyzing dynamic adversarial training data in the limit.

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

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W. Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. 2018. HotpotQA: A dataset for diverse, explainable multi-hop question answering. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Victor H. Yngve. 1960. A model and an hypothesis for language structure. *Proceedings of the American Philosophical Society*, 104(5):444–466.

Tao Yu, Rui Zhang, Kai Yang, Michihiro Yasunaga, Dongxu Wang, Zifan Li, James Ma, Irene Li, Qingning Yao, Shanelle Roman, Zilin Zhang, and Dragomir Radev. 2018. Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-SQL task. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3911–3921, Brussels, Belgium. Association for Computational Linguistics.

A Appendices

A.1 Dataset Statistics

Refer to Table 6 for final split sizes. When creating the splits, we first partition the available data into train and evaluation (combined size of dev + test) sets using the methodology of each split (e.g. TMCD maximizes compound divergence, Likelihood Splits sort by an LM score and then partition the data). Then the evaluation set is randomly divided into dev and test sets.

Dataset	|Train|	|Dev|	|Test|	Total
SPIDER	5,966	1,034	1,034	8,034
SNLI	549,018	10,000	10,000	569,018
BOOLQ	7,617	2,540	2,540	12,697

Table 6: Sizes of train/dev/test sets for the dataset splits

The public release of the SPIDER dataset consists of 7000 training examples and 1034 validation examples (it also contains 1659 additional examples from older datasets which we do not use in our work). We use these 8034 examples to create all our splits. Shaw et al. (2021), one of the alternative splits that we compare against, use a subset of 4000 examples from the 7000 training examples. Hence, our results are not directly comparable to performance reported by them. The SQL programs for 6 (of 8034) examples in the dataset cannot be parsed uniquely and thus we cannot define compounds on these examples. We drop these examples when creating the TMCD split i.e. the training set of TMCD split contains 6 fewer examples.

The SNLI data contains 550152 training examples, 10000 dev examples and 10000 test examples for a total of 570152 examples. We drop examples where the gold label cannot be determined by majority vote. We also drop examples where the premise was labelled ‘Cannot see picture to describe.’ or the hypothesis is empty. This results in a filtered dataset of 569018 examples from which we create our splits.

The public release of BOOLQ contain 9427 labeled training examples, 3270 labeled development examples, and 3245 unlabeled test examples. Thus we create our splits from the 12,697 labelled train and development examples. We maintain the approximate 60/20/20 train/dev/test proportions of the original dataset when creating the splits.

A.2 Model Hyperparameters

We use the Transformers library (Wolf et al., 2020) for training and evaluation. All models were trained on Nvidia Quadro RTX 6000 GPUs (24GB GPU Memory).

We report hyperparameters for fine-tuning GPT-2 to create the Likelihood Splits in Table 7. We select the best checkpoint based on lowest perplexity by validating on 10% of the training data in each fold.

Hyperparameters for SPIDER models are in Table 8, for SNLI models in Table 9 and for BOOLQ models in Table 10. Note that we evaluate checkpoints (see hyperparameter ‘eval steps’) during training to select the best checkpoint at the end of training.

Hyperparameter	Value
train batch_sz	32
lr_scheduler	constant
learning_rate	2e-5
optimizer	AdamW
eval steps	64
max steps	SPIDER: 2000 SNLI: 15000 BOOLQ: 3000

Table 7: Hyperparameters for fine-tuning GPT-2 (both medium and large) on the dataset folds to create Likelihood Splits

	T5	GPT-2
train batch_sz	8	2
grad acc steps	16	16
max_steps	10000	10000
lr_scheduler	constant	constant
learning_rate	1e-3	2e-5
optimizer	Adafactor	AdamW
max src_length	512	512
max tgt_length	256	256
src prefix	-	`database question for table`
tgt prefix	`semanticparse:`	`generate the sql parse:`
eval batch_sz	8	1
eval steps	256	128
num_beams	5	5

Table 8: Hyperparameters for the models trained on SPIDER.

For SPIDER, we follow Shaw et al. (2021) and tune the learning rate, batch size and maximum training steps for a T5-base model (Raffel et al., 2020) on a random split of the SPIDER dataset. Once we have found a hyperparameter setting, we apply the same setting on the all splits. We also report performance of a T5-small model on all splits trained with the same hyperparameters.

For SNLI and BOOLQ, we follow the default hyperparameters suggested by the original works. Additionally, we perform early stopping when performance on the validation set fails to improve for a specified number of evaluations.

A.3 Effect of k on Likelihood Splits

When creating Likelihood Splits, the number of folds k for fine-tuning (§3.2) is a choice left to the creator of the benchmark. In our work, we

	RoBERTa	ELECTRA	
		base	large
train batch_sz	32		
max seq length	128		
lr_scheduler	linear		
optimizer	AdamW		
adam_beta1	0.9		
adam_beta2	0.98	0.999	
adam_epsilon	1e-6		
num epochs	10	3	
warmup ratio	0.06	0.1	
layer. lr decay	1.0	0.8	0.9
learning_rate	1e-5	1e-4	5e-5
weight decay	0.1	0.0	
eval batch_sz	32		
eval steps	256		
patience	20	n/a	

Table 9: Hyperparameters for the models trained on SNLI. Patiences refer to number of evaluations with no improvement before early stopping.

	RoBERTa	ELECTRA	
		base	large
train batch_sz	8		
grad acc steps	4		
max seq length	512		
lr_scheduler	linear		
optimizer	AdamW		
adam_beta1	0.9		
adam_beta2	0.98	0.999	
adam_epsilon	1e-6		
num epochs	10	5	
warmup ratio	0.06	0.1	
layer. lr decay	1.0	0.8	0.9
learning_rate	1e-5	1e-4	5e-5
weight decay	0.1	0.0	
eval batch_sz	8		
eval steps	128		
patience	10	n/a	

Table 10: Hyperparameters for the models trained on BOOLQ. Patience refers to number of evaluations with no improvement before early stopping.

set $k = 3$ as an arbitrary choice prior to running the task models i.e. it was not tuned based on task model performance. We conduct additional experiments to test the effect of changing the value of k by setting $k = 5$ and generating new splits.

On SPIDER, when $k = 5$ instead of $k = 3$, Likelihood scores are highly correlated with a Pearson's r of 0.90. However, the process of balancing atoms (described in §3.4) causes the evaluation sets to look more different. When controlling for length, 77% of the evaluation set examples are the same; 80% of the evaluation examples are the same otherwise. We report the accuracy of T5-base (the most competitive baseline model on SPIDER) on the new splits with $k = 5$ in Table 11. The new splits are more difficult by about 2%. We observe the same trend where controlling for length makes the splits less challenging.

On BOOLQ, when using 5 folds instead of the 3 folds, Likelihood scores are highly correlated with a Pearson's r of 0.96 and 89% of the evaluation set examples are the same. Accordingly, the ELECTRA-large accuracy only changes slightly from 82.6% to 83% on the new test set and is still lower than the random split accuracy. While there exists an indication that controlling for length makes the *ll_split (-len)* splits more challenging, the effect of controlling for length becomes more pronounced when $k = 5$. We report the effect of k on BOOLQ performance in detail in Table 12.

We report the effect of changing k on SNLI accuracy in Table 12. Since the SNLI dataset is an order of magnitude larger than the SPIDER and BOOLQ datasets, the number of folds has less of an impact on the LM fine-tuning. As a result, Likelihood scores for $k = 3$ and $k = 5$ are highly correlated with a Pearson's r of 0.99. When controlling for length, 89% of the evaluation set examples are the same; 92% of the evaluation examples are the same otherwise. Accordingly, we see much smaller differences in model performance on the new splits; the ROBERTA model accuracies change by at most 0.9% on the new test sets.

	SPIDER				
	Random	ll_split (k=3)		ll_split (k=5)	
System			(-len)		(-len)
T5-base	78.6	66.0	71.3	64.8	69.2

Table 11: Effect of k on the difficulty of Likelihood Splits of SPIDER. The accuracies of T5-base on the Likelihood Split are comparable and significantly lower than the accuracy on Random split. Controlling for length decreases the difficulty in both cases.

A.4 SPIDER: Variation of TMCD Related Properties

Past work by Keysers et al. (2020) has established the terms of atom and compound "divergence" to quantitatively describe the extent to which the distributions of the atoms and compounds differ between the train and evaluation sets. They used the Chernoff coefficient (Chung et al., 1989) to measure distribution similarity:

	SNLI					BOOLQ				
	Random	ll_split (k=3)		ll_split (k=5)		Random	ll_split (k=3)		ll_split (k=5)	
System			(-len)		(-len)			(-len)		(-len)
ROBERTA-base	89.6 ±0.4	79.3	77.1	78.4	75.0	74.9 ±0.4	71.6	71.2	71.1	70.3
ROBERTA-large	90.5 ±0.5	82.3	79.2	82.1	79.0	84.4 ±0.6	79.3	78.9	78.4	78.3
ELECTRA-base	90.5 ±0.2	80.1	78.3	80.1	77.6	78.8 ±1.1	74.1	74.3	75.0	73.9
ELECTRA-large	91.0 ±1.3	82.6	81.6	84.0	80.6	85.5 ±0.6	82.6	82.1	83.0	80.6
Human Accuracy	88.7 ±0.8	83.6	84.4	83.0	84.0	-	-	-	-	-

Table 12: Effect of k on the difficulty of Likelihood Splits of SNLI and BOOLQ. There are some accuracy differences on BOOLQ, however the values are comparable and significantly lower than the accuracy on Random splits. The accuracy differences are less pronounced on SNLI.

$$C_\alpha(P \parallel Q) = \sum_k p_k^\alpha q_k^{1-\alpha} \qquad \in [0,1] \qquad (1)$$

where p_k and q_k are the probability of a particular atom/compound k being in the train and test set respectively. The divergence is then $1 - C_\alpha$. The “atom" divergence uses $\alpha = 0.5$ as a symmetric divergence score. The “compound" divergence used $\alpha = 0.1$ to give more importance to the occurrence of a compound in the train set rather than trying to make the distributions of train and test set similar.

Split	Atom	Compound
Random	0.077	0.046
Length	0.120	0.092
Template	0.105	0.089
TMCD	0.296	0.322
ll_split	0.083	0.054
ll_split (-len)	0.081	0.049
ll_split prompt	0.093	0.056
ll_split prompt (-len)	0.094	0.052

Table 13: Atom and Compound divergence (on the logical form side) between train and dev sets of various splits. Although we ensure every atom appears at least once in the train set, a high atom divergence demonstrates the challenge of learning rare atoms. A greater than random compound divergence emerges denoting a need for compositional generalization.

A.5 SPIDER: Variation of SQL Hardness

We use a tool provided by the SPIDER dataset creators to evaluate hardness. The tools assigns a rating from easy, medium, hard or extra hard to every example based on the complexity of the SQL parse. Complexity is evaluated in terms of the number of join and aggregation operations, and nested SQL statements. We find that the Likelihood Splits are skewed towards putting more complex examples in the evaluation set compared to the test set (see Figure 5).

A.6 SPIDER: Input Length Variation

As expected, the likelihood assigned by the language model (LM) is negatively correlated with sequence length meaning i.e. longer sequences tend to have lower likelihood. This can be seen from Figure 6, where *ll_split* and *ll_split pt* tend to put longer utterances in the lower likelihood evaluations set. Accounting for length by dividing the data within buckets makes the distribution of train and test sets align better and remove the added difficulty of length generalization. The *length* split poses this challenge which has been established to be a difficult ability for generation models to acquire (Newman et al., 2020). Note that the distributions do not match exactly since examples need to be swapped between train and evaluation set to meet the atom constraint (evaluation cannot contain any unseen atoms).

A.7 SPIDER: Variation of Query Parse Structure

We analyze the complexity of the parse structure of the queries. Following Wallace et al. (2021), we parse the queries using the Benepar parser (Kitaev and Klein, 2018) based on T5 small (Raffel et al., 2020). We report the distributions of mean and max parse tree depth as well as the syntactic complexity of the utterance based on the Yngve score (Yngve, 1960; Roark et al., 2007). The Yngve score essentially measures the average number of left branches on the path from the root of the parse tree to every word in the sentence and can be thought of as measuring the number of spans that need to be coordinated.

We can see that the dev set of the *ll_split* is on average more complex than its train set along all 3

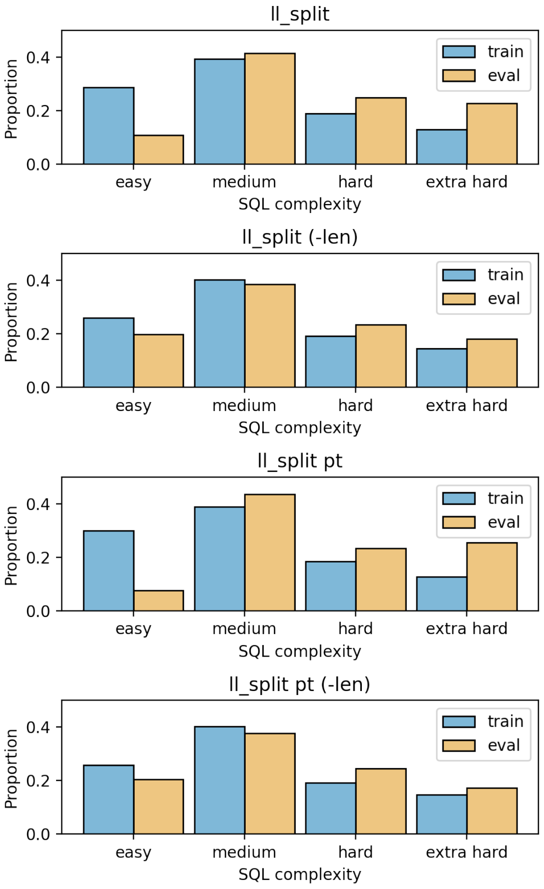

Figure 5: SPIDER: Distribution of SQL programs of varying complexity in the train and development set of various splits. Likelihood Splits show a skew towards training on easy examples and evaluating on harder examples.

metrics considered (see Figure 8). Moreover, these metrics are correlated with utterance length, and controlling for it in the *ll_split (-len)* split makes the difference less pronounced.

A.8 SPIDER: Error Analysis

We analyze the performance of T5-base on the development set of *ll_split pt*. In particular, we test whether the presence of novel compounds and SQL query hardness are sufficient to explain the difficulty.

We call compounds in the SQL programs of the development set as 'novel' if they do not occur in the training set of the split. 25.5% of the dev set examples in the random split contain at least one novel compound as opposed to 43.6% of the dev set examples of *ll_split pt*. From Table 14, we see that T5-base performance is lower in both categories of examples. Projecting for expected performance on dev set of *ll_split pt* assuming the examples were as difficult as examples from the random split over-estimates the performance of T5-base.

Split	Random	*ll_split pt*
Percent. of examples with a novel compound	25.5%	43.6%
Acc on examples with novel compounds	61.4%	45.5%
Acc on remaining examples	87.1%	79.8%
Acc on the dev set	80.6%	64.8%
Projected accuracy		75.9%

Table 14: SPIDER: The presence of novel compounds alone does not explain the difficulty of the *ll_split pt*. Projecting the random set accuracies using the percentage of examples with novel compounds in *ll_split pt* over-estimates dev set performance.

We report performance of T5-base on the dev sets grouped by the SQL hardness metric (described in Appendix A.5) in Table 5. We see that accuracy on *ll_split pt* is lower than the accuracy on the random set within each SQL complexity bucket. If the sole source of difficulty was the larger proportion of harder examples, projecting the random set accuracies would correctly estimate dev set performance on *ll_split pt*. However, the projection is an over-estimate. Thus, the hardness metric alone does not explain the difficulty of the proposed split.

A.9 SNLI: Input Length Variation

From Figure 9, we see that the Likelihood Splits put longer premises and hypotheses in the evaluation set. Controlling for length completely removes this skew while increasing the difficulty of the splits (Table 3). This means that if we remove the factor of length from likelihood, the remaining examples have lower likelihood for other reasons; reasons that contribute to difficulty.

A.10 SNLI: Distribution of Rare Words

We report the fraction of test sets words that are rare for various splits in Figure 7. This evaluation combines the premise and the hypothesis i.e. it considers the full task input. In order to remove typographical errors, we only consider words that occur in a wordlist of English words (`https://github.com/dwyl/english-words`). We define rare words as words that occur at most 1 time per million words statistics collected in SUBTLEXus (Brysbaert and New, 2009). This process results in a list of 13478 low frequency words that occur in the SNLI dataset. We find that Likelihood Splits put examples with a significantly large fraction or rare words in the evaluation set. Controlling for length increases the fraction of rare words since length is removed as a factor from likelihood.

A.11 SNLI: Variation of Syntactic complexity

We compute Yngve scores for premise and hypothesis of examples as described in Appendix A.7. The complexity of premise and hypothesis in developments sets of Likelihood Splits is higher than in the corresponding train sets (see Figure 10). Controlling for length removes this skew in the premise. However, the length controlled splits tend to have less syntactically complex hypotheses in the evaluation sets. This is surprising because the length-controlled variants are actually more difficult for the model; human performance is higher on length-controlled splits.

A.12 SNLI: Variation of Reading Level

We compute the Flesch-Kincaid reading level (Kincaid et al., 1975) for premise and hypothesis of examples. This score takes into account the number of syllables per word in the sentence. The reading grade (complexity) of premise and hypothesis in developments sets of Likelihood Splits is higher than in the corresponding train sets (see Figure 11). Controlling for length does not fully remove this skew and the evaluation examples retain more complex sentences than in the training set.

A.13 SNLI: Error Analysis

In Table 15, we present some examples from the development set of *ll_split (-len)* where the fine-tuned ROBERTA-large model predicts incorrectly. We divide them into categories: examples requiring external world knowledge, examples where a typo changes the meaning of the example, and examples with ambiguous or incorrect labels.

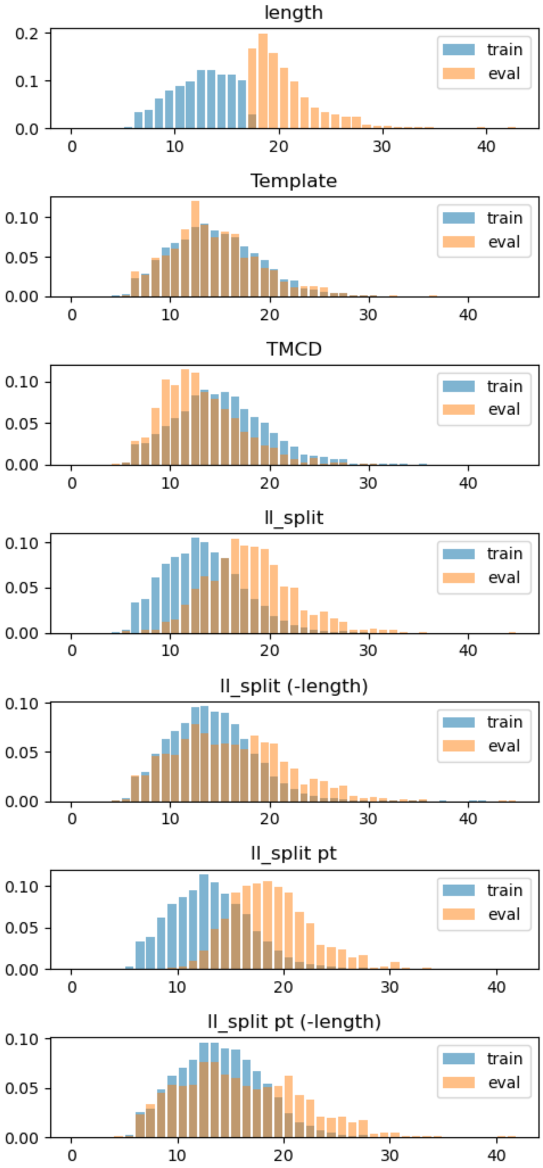

Figure 6: SPIDER: Input length variation for the splits. Y-axis is the distribution of examples within each length bucket of the X-axis

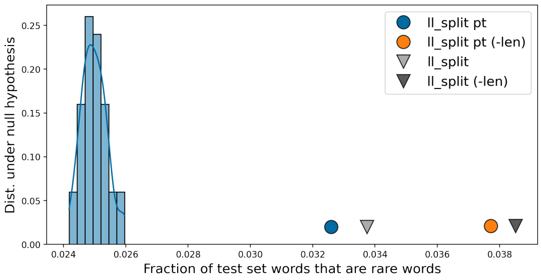

Figure 7: SNLI: Fraction of development set words that are rare in the premise and hypothesis of various splits. The dev sets for *ll_split* seem to contain a larger fraction of rare words than random splits. Normalizing for the length seems to retain more rare words.

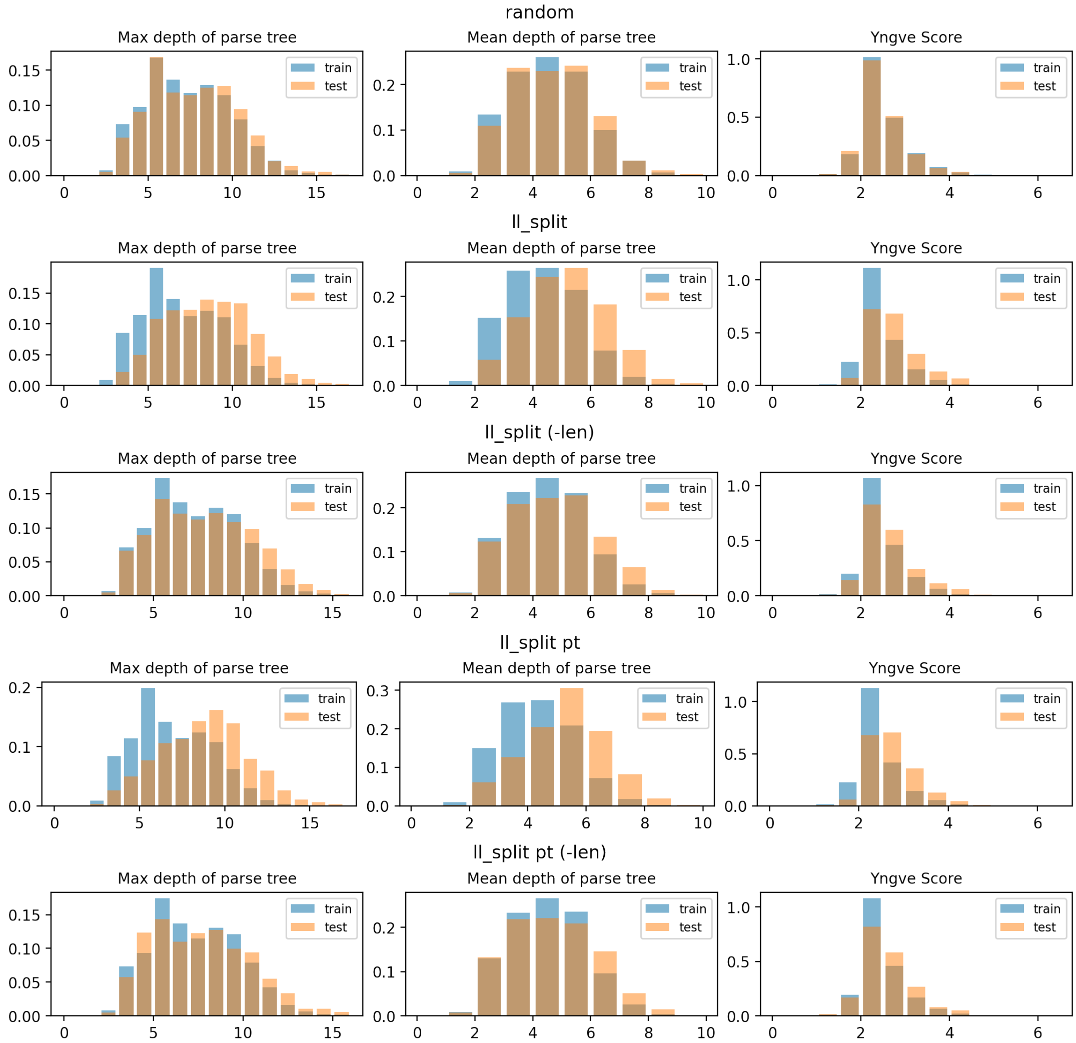

Figure 8: SPIDER: Distribution of features computed on the parse tree of the input query. The dev sets for *ll_split* seem to contain more complex utterances across all 3 metrics considered. Normalizing for the length seems to reduce the effect. The metrics are mean and max depth of parse tree and Yngve score which is a metric

Error Type	Premise	Hypothesis	Gold Label	Predicted Label
Requires External Knowledge	A young boy is holding on and riding a zip line down a hill.	A exciting adventure!	entailment	neutral
	A young child is watching a toy construction brick construct.	A child is using lincoln logs.	neutral	contradiction
	A performer is jumping off the stage into a crowd of fans.	The artist is crowdsurfing.	entailment	neutral
	A couple holds up their child on a series of large steps while others are also traversing the steps.	A fourteen year old is restrained from the museum.	contradiction	neutral
Typo	Martial artists perform in front of a crowd outdoors.	There is a crown outdoors.	entailment	neutral
Ambiguous / Incorrect Labels	Technicians working in underground.	People work underground while dinosaurs attacked	neutral	contradiction
	A young gentlemen with a blue tie talking into a microphone.	High winds will interfere with microphone recording.	entailment	neutral

Table 15: SNLI: Error analysis of ROBERTA-large on examples from *ll_split (-len)*.

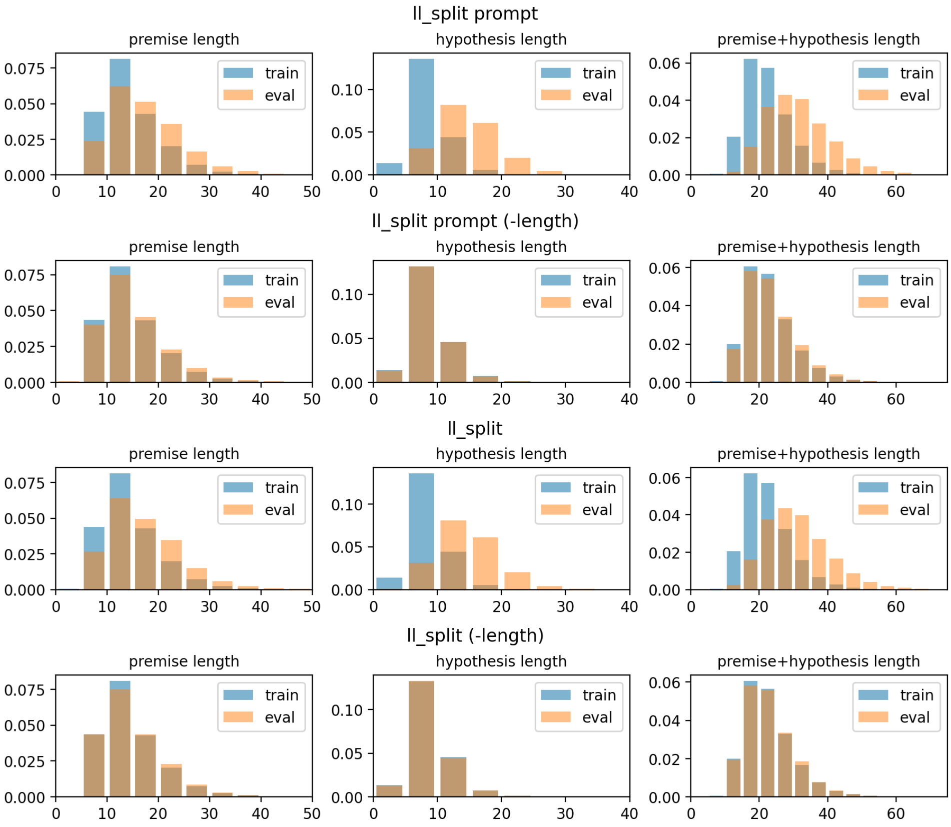

Figure 9: SNLI: Input length variation of premise and hypothesis for the splits.

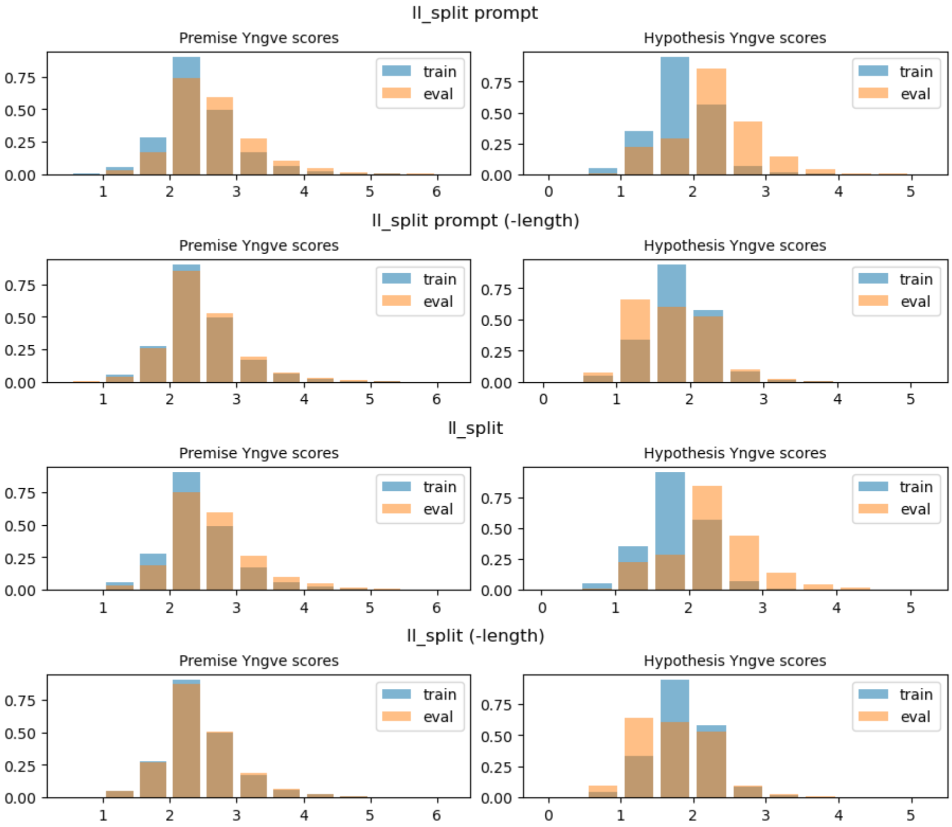

Figure 10: SNLI: Distribution of Yngve scores (syntactic complexity) computed on the parse tree of the premise and hypothesis. The dev sets for Likelihood Splits contain more complex utterances.

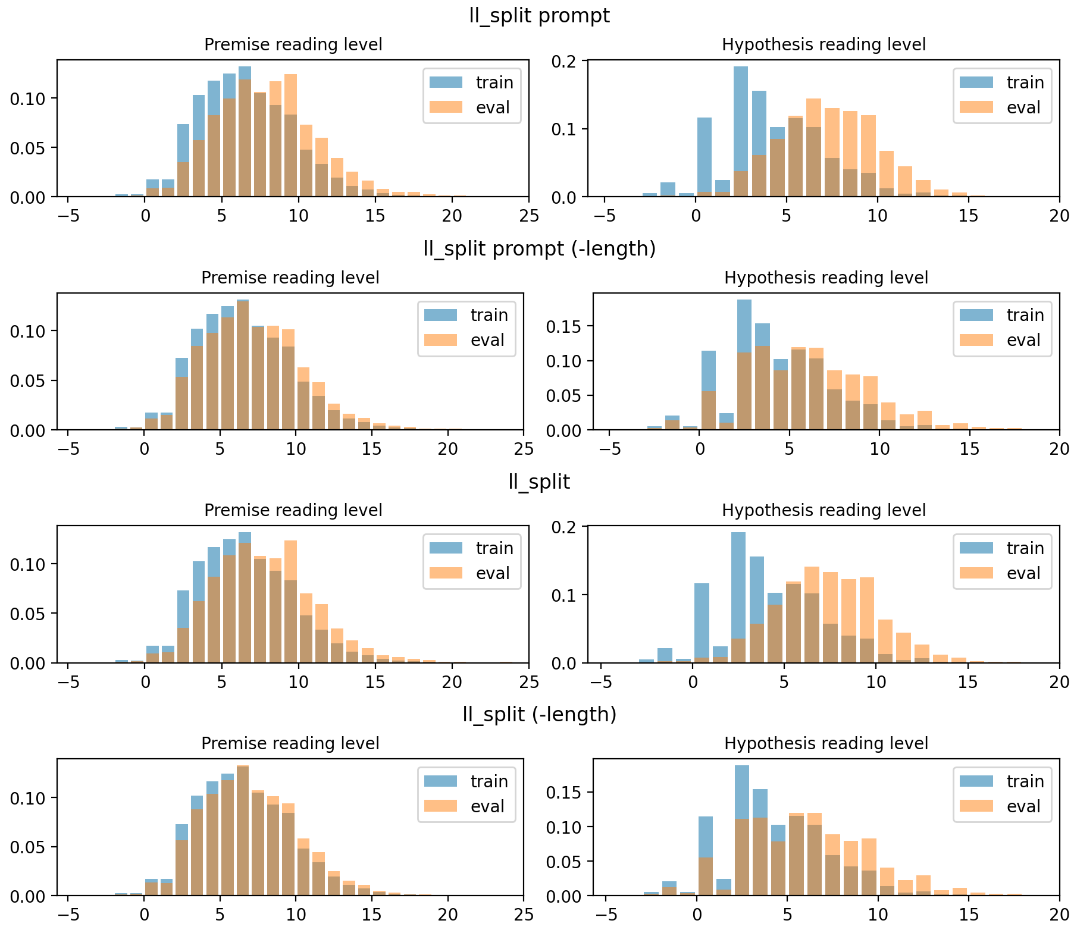

Figure 11: SNLI: Distribution of Flesch-Kincaid reading level for the premise and hypothesis in various splits. The dev sets for *ll_split* seem to contain more complex utterances. Normalizing for the length seems to reduce but not remove the effect.