Quantifying Adaptability in Pre-trained Language Models with 500 Tasks

Belinda Z. Li
MIT
bzl@mit.edu

Jane Yu
Meta AI
janeyu@fb.com

Madian Khabsa
Meta AI
mkhabsa@fb.com

Luke Zettlemoyer
Meta AI
lsz@fb.com

Alon Halevy
Meta AI
ayh@fb.com

Jacob Andreas
MIT
jda@mit.edu

Abstract

When a neural language model (LM) is adapted to perform a new task, what aspects of the task predict the eventual performance of the model? In NLP, systematic features of LM *generalization* to individual examples are well characterized, but systematic aspects of LM *adaptability* to new tasks are not nearly as well understood. We present a large-scale empirical study of the features and limits of LM adaptability using a new benchmark, TASKBENCH500, built from 500 procedurally generated sequence modeling tasks. These tasks combine core aspects of language processing, including lexical semantics, sequence processing, memorization, logical reasoning, and world knowledge. Using TASKBENCH500, we evaluate three facets of adaptability, finding that: (1) adaptation procedures differ dramatically in their ability to memorize small datasets; (2) within a subset of task types, adaptation procedures exhibit *compositional adaptability* to complex tasks; and (3) failure to match training label distributions is explained by mismatches in the intrinsic difficulty of predicting individual labels. Our experiments show that adaptability to new tasks, like generalization to new examples, can be systematically described and understood, and we conclude with a discussion of additional aspects of adaptability that could be studied using the new benchmark.

1 Introduction

Much of the recent research effort in NLP has shifted from training task-specific models to *adapting* pre-trained language models (LMs) by fine-tuning their parameters or input prompts for downstream tasks (Devlin et al., 2019; Raffel et al., 2020; Li and Liang, 2021; Lester et al., 2021). This paradigm is general, in the sense that a large number of distinct NLP tasks benefit from pre-training (Peters et al., 2018; Devlin et al., 2019; Raffel et al., 2020). But many questions about the nature and limits of LM adaptation remain unanswered. For example: given a new task, can we predict how quickly (and how effectively) pre-trained LMs can be adapted to perform it? From among the variety of different adaptation techniques (e.g. fine-tuning or prompt-tuning), can we predict which one will be most effective? Today, new pre-training and adaptation schemes are evaluated using small suites of curated tasks, typically featuring classification, textual inference, and question answering (Wang et al., 2018, 2019). These benchmarks have been extremely successful in identifying new tools for adaptation, but they are poorly suited for answering larger, structural questions like the ones above.

We present a large-scale study of LM adaptability using a new suite of benchmark tasks called TASKBENCH500. [1] TASKBENCH500 consists of 500 **procedurally generated** tasks involving lexical semantics, factual information, memorization of random relations, list processing, and logical composition (Fig. 1). Analogous to past work that uses synthetic data to characterize LM performance on single examples (Weston et al., 2016; Lake and Baroni, 2018; Saxton et al., 2019; Kim and Linzen, 2020; Keysers et al., 2020; Liu et al., 2021a), TASKBENCH500 enables systematic study of LM adaptability at the task level. In this paper, we use it to study three aspects of adaptability:

Memorization: When can adaptation successfully memorize new functions (e.g., to update factual knowledge about entities, or learn arbitrary new token correspondences)? We find that **LMs' ability to memorize new input–output mappings is strongly influenced by task type**. Datasets of lexical relations (like antonym pairs) are easier to memorize than factual information (like name–occupation pairs). Both are easier to memorize than lists of random word pairs. These findings are particularly striking in the case of prompt tun-

[1] Data and code available at: https://github.com/facebookresearch/task_bench

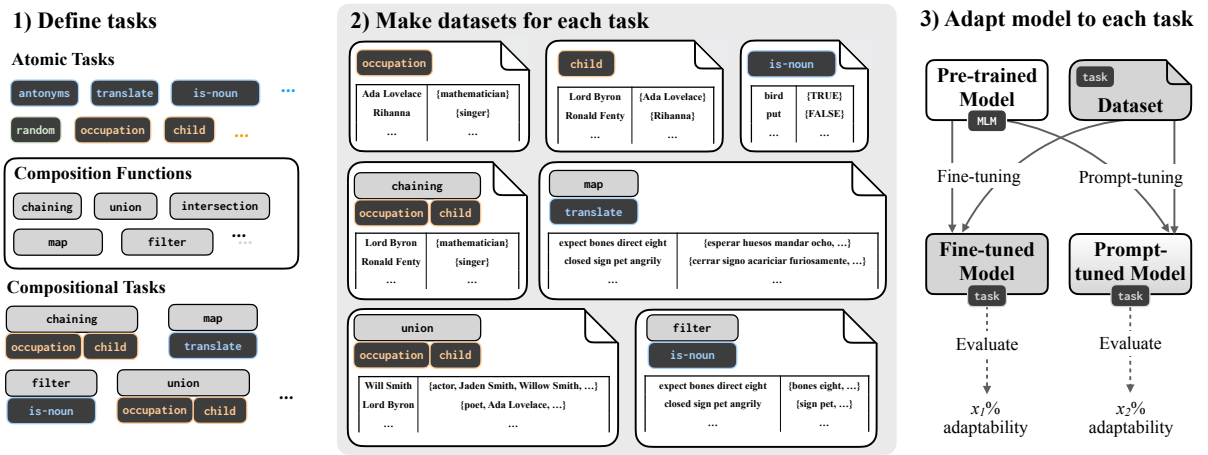

Figure 1: Overview of our task creation process. We begin by defining a set of atomic tasks that all synthetic tasks are built upon. These include lexical tasks (blue text/outline), random tasks (green text/outline), and factual tasks (orange text/outline). They also include both predicates and relations. These tasks are combined using composition functions to form more complex, compositional tasks. Given a particular task specification, we synthetically create a dataset for each task. Finally, we fine-tune or prompt-tune a pre-trained language model on each task dataset.

ing, which in standard configurations struggles to memorize even small random word pair lists.

Composition: Is LM performance on simple tasks predictive of their performance on compositions of those tasks? (If the *father* and *occupation* relations are easy to learn via adaptation, does this imply that the *father's occupation* relation is also easy to learn?) We find a nuanced answer. **LMs exhibit compositional adaptation** to lexical and factual relations (like *father's occupation*), with success on composed tasks strongly correlated ($r^2 > 0.5$) with success on atomic tasks. However, when composing these relations with sequence processing operations, success on the base task does not predict success on the composed task.

Distribution matching: In models fine-tuned on datasets exhibiting a distribution of acceptable answers (e.g., translating ungendered pronouns into gendered ones), do model predictions match these distributions? We find that **LMs are often unable to match label distributions in datasets used for adaptation**. In particular, when labels in the fine-tuning dataset are drawn from a uniform mixture of labels from two tasks (e.g., labeling half of the words with their *antonym* and half with their *synonym*), models disproportionately assign mass to labels from the task that is easier to learn.

Each of these forms of adaptability corresponds to a central challenge in NLP: reliable updating of deployed models, composition of previously learned skills, and fair and predictable output from models trained on curated data. Our study of memorization, composition, and distribution matching have direct analogs to previous studies of sample expressivity (Zhang et al., 2017), compositional generalization (Lake and Baroni, 2018; Kim and Linzen, 2020; Keysers et al., 2020), and calibration (Guo et al., 2017). However, we study these phenomena at the task level, rather than the example level. Our experiments highlight important qualitative differences between current adaptation paradigms; identify several novel challenges for LM adaptation, and offer a new benchmark for approaches aimed at meeting these challenges.

2 Background

Fine-tuning and prompt search In languages for which large digitized corpora are available, most NLP system development today involves *adaptation* of a pre-trained model to a downstream task of interest. Pre-training typically involves reconstruction of masked or corrupted text sampled from a large corpus (Devlin et al., 2019; Liu et al., 2019; Raffel et al., 2020). Adaptation to a new task typically involves one of three approaches: (1) **fine-tuning** of all of a pre-trained model's parameters (possibly in conjunction with a specialized decoder) on a task-specific training set (Devlin et al., 2019); (2) manual **prompt engineering** of an input template that induces pre-trained model predictions to perform the task of interest (Brown et al., 2020; Petroni et al., 2019); or (3) automated **prompt tuning** of these templates, in either the discrete space of tokens (Shin et al., 2020) or con-

tinuous space of token embeddings (Li and Liang, 2021; Lester et al., 2021; Liu et al., 2021b). The latter two approaches have grown more popular as pre-trained models have grown larger. The performance of both prompt-search approaches still lags fine-tuning (Raffel et al., 2020; Brown et al., 2020; Lester et al., 2021), though the difference between approaches appears to shrink as model size increases (Lester et al., 2021).

Measuring generalization and adaptability The success of the training paradigm described above stems from its generality—a large number of NLP tasks appear to benefit from some combination of pre-training and adaptation. Previous attempts to *quantify* this generality have typically relied on benchmarks like GLUE (Wang et al., 2018) and SuperGLUE (Wang et al., 2019), each of which aggregates ten natural language processing tasks designed to probe different aspects of language understanding. Similar benchmarks have also been built for non-English languages (Xu et al., 2020; Kakwani et al., 2020; Park et al., 2021; Hu et al., 2020). However, the heterogeneity and small number of distinct tasks represented in existing benchmarks makes it difficult to make finer-grained predictions, e.g. by identifying specific features of tasks that contribute to the success or failure of adaptation.

This challenge has a direct analog to the problem of characterizing *generalization* at the example level in models trained for a single task. Model performance on natural test sets is often loosely correlated with accuracy on individual examples featuring rare syntactic constructions or word collocations (McCoy et al., 2019). A great deal of past work has focused on improving evaluation using synthetic evaluation sets (Jia and Liang, 2017; Naik et al., 2018; Lake and Baroni, 2018; Richardson et al., 2020). These datasets have been used to study long-range agreement (Marvin and Linzen, 2018), compositional generalization (Lake and Baroni, 2018; Ruis et al., 2020; Keysers et al., 2020), and mathematical reasoning (Saxton et al., 2019). But no analogous notion of systematicity, or tool for studying it, currently exists for tasks rather than examples.

Thus, building on this past work, we describe how to construct synthetic data distributions that enable systematic study of *adaptation to new tasks* rather than *generalization to new examples*. Like previous research that uses procedural data generation procedures to study models in NLP, we focus on coverage rather than naturalness, using datasets designed to complement, rather than replace, existing naturalistic benchmarks.

3 A 500-task benchmark

Our goal is to study the generalizability of task adaptation paradigms. In particular, we would like to identify which attributes of a task make it easy or difficult to learn, across different models and training schemes. While this work shares many of its high-level goals with existing benchmarks built from collections of real-world datasets, the makeup and difficulty of these datasets is often difficult to characterize precisely: differences in annotation standards, annotation quality, and dataset size mean that models often exhibit very different performance on datasets designed to evaluate model performance on the same abstract task. In addition, existing datasets cover an exceedingly small subset of the space of all tasks that future NLP practitioners might wish to perform. To account for all these limitations, we propose to generate tasks *synthetically* as described below.

The space of tasks TASKBENCH500 is constructed compositionally: we begin by defining a space of **atomic tasks**, which are combined using a set of **composition operators** to produce more complex tasks. Every task takes as input a word or word sequence, and outputs either a boolean value or a set of words/word sequences. We refer to any task that outputs booleans as a **predicate task**, and any task that outputs sets of words or word sequences as a **relation task**. A subset of relation tasks involve modeling relations between single words at the input and output; we refer to these as **word-level** tasks and the remaining relation tasks (that take sequences as input or output) as **sequential tasks**.

The choice of atomic tasks and composition functions aims to capture aspects of real language processing tasks. Accordingly, the set of atomic tasks comprises of:

1. **Lexical tasks**, which test knowledge of lexical semantics. These include *lexical relations* like `synonym`, or *lexical predicates* like `is-noun`. These tasks are constructed from WordNet relations (Fellbaum, 1998).

2. **Factual tasks**, which test factual knowledge. These include *factual relations* like `father-of`, or *factual predicates* like

`is-human`. These tasks are constructed from Wikidata properties (Vrandečić and Krötzsch, 2014).

3. **Random relation tasks**, which test memorization ability. These are created by mapping a word in the vocabulary to a singleton set containing a random other word. We create 4 random relations with different random seeds.

To recursively create arbitrarily complex tasks, we define a set of *composition functions*. These take tasks as arguments and return other tasks. These functions fall into two categories:

1. **Word-level compositions**, which test ability to combine word-level information in different ways, such as through set or logical operations. These functions take word-level tasks and return other word-level tasks. Examples include `intersection` and `chaining`.

2. **Sequential compositions**, which test ability to operate on sequences. These functions convert word-level tasks to sequence-level tasks. There are two functions in this category: `map` takes a word-level relation task and returns a task that maps a sequence of n words to a set of all possible sequences resulting from applying f_W to each input word.[2] `filter` takes word-level predicate tasks and returns a sequence consisting only of words for which the task returns `true`, preserving the original ordering of those words.

The full list of atomic tasks and composition function can be found in Appendix Tables 4 and 5. We surmise that typical NLP tasks may require some combination of lexical knowledge, factual knowledge, sequential processing, and other task-specific reasoning; our data distribution lets us evaluate all these aspects separately and in combination.

Datasets for tasks We create datasets $\mathcal{D}(f) = \{(x_i, y_i) : x \sim \mathcal{X}_f, y \sim \text{Unif}(f(x_i))\}$ for each task f, where $\mathcal{X}_f$ is the input distribution for the task, and recalling that $f(x_i)$ returns a *set* of possible outputs associated with the input x_i. For all tasks, we randomly partition the dataset into $\mathcal{D}_{\text{train}}(f)$ and $\mathcal{D}_{\text{test}}(f)$ splits.

[2]Note word-level relations return *sets* of words—we turn a sequence of sets of words into a set of sequences by considering all combinations of words in each set.

For *lexical atomic tasks* and their compositions, we directly use the most common words in the task's input language for $\mathcal{X}_f$. We create tasks in English and Spanish. For *factual atomic tasks* and their compositions, we sample the entities from Wikidata that participate in the relation or predicate defined by the task (e.g. for the `child-of` task, we sample only entities that have children). For *sequential tasks*, we use a random sampler, which samples n random words from the vocabulary and concatenates them.

Figure 1 shows examples of tasks and associated datasets. More details on dataset construction can be found in Appendix A.

4 Experimental Setup

Model & Training For all experiments, we adapt a pre-trained T5-base model (Raffel et al., 2020). We examine two types of training paradigms: fine-tuning and prompt-tuning. During fine-tuning, we update all model parameters on the training set. During prompt-tuning, we follow Lester et al. (2021) and introduce a new set of prompt-tokens $\{p_1, \cdots, p_n\}$ to the vocabulary, which will be prepended to every sample from the task during inference, i.e., each sample input x becomes $p_1 p_2 \cdots p_n x$. Let $\boldsymbol{\theta}$ denote the parameters of the original pretrained LM. During training, the entire model is frozen and only the word embeddings of the new tokens $\{\theta_{p_1}, \cdots, \theta_{p_n}\} \subset \boldsymbol{\theta}$ are updated. We use prompts of length $n = 100$ for all experiments. We also study each paradigm on various quantities of training data, and separately evaluate their memorization and generalization adaptabilities. In particular, for word-level tasks the test-set words are disjoint from the train-set words, so evaluating on the test set will strictly measure generalization capacity. We optimize all models using AdamW. See Appendix B for full hyperparameters.

Evaluation For each task f and model $\mathcal{M}[\boldsymbol{\theta}]$ (with parameters $\boldsymbol{\theta}$), we measure the model's average per-token accuracy on both training and test splits of the dataset $\mathcal{D}(f)$. As each task defines multiple acceptable outputs for each input, we credit models for producing any acceptable output. Letting $y' = \mathcal{M}(x)$, we measure the fraction of positions i at which any valid answer y_i matches the

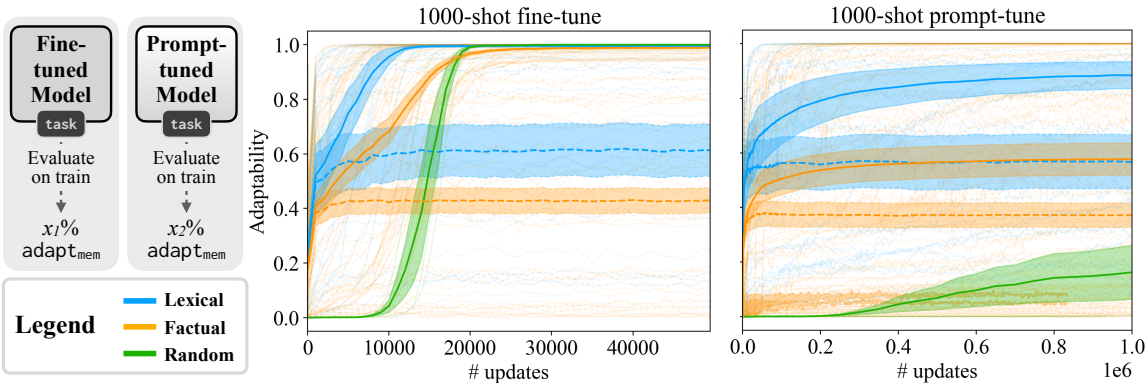

Figure 2: Left: Overview of the memorization experiment, which evaluates how accurately models adapted via fine-tuning and prompt-tuning can memorize training data. Right: Memorization and generalization curves for fine-tuning and prompt-tuning on 1000 training examples. Memorization curves are shown by solid lines, while generalization curves are dashed. We average over all atomic tasks from each task category: lexical tasks, factual tasks, and random tasks. The shaded region shows the standard error of the mean. Transparent lines are each individual task, colored by task category. In both paradigms, lexical tasks are easiest to memorize, followed by factual tasks, then random tasks. However, prompt-tuning has overall much less memorization capacity than fine-tuning, which can perfectly memorize even completely random relations.

predicted y_i':

$$\texttt{acc}(\mathcal{M}, \mathcal{D}(f)) = \max_{y \in f(x)} \frac{1}{|\mathcal{D}|} \sum_{(x,y)\in\mathcal{D}} \left(\frac{1}{n} \sum_{i=1}^{n} [\![y_i = y_i']\!] \right) \quad (1)$$

Further details can be found in Appendix B.

Given a pretrained model $\mathcal{M}[\boldsymbol{\theta}_{\texttt{pretrain}}]$, an adaptation procedure $\mathcal{T}$, and a task suite f, let $\mathcal{M}[\boldsymbol{\theta}_{\mathcal{T},\mathcal{D}(f)}]$ denote the model trained using $\mathcal{T}$ on training data $\mathcal{D}(f)$. We then define the *adaptability* of a (pretrained model, adaptation paradigm, task suite) as:

$$\texttt{adapt}(\mathcal{M}[\boldsymbol{\theta}_{\texttt{pretrain}}], \mathcal{T}, f) = \texttt{acc}(\mathcal{M}[\boldsymbol{\theta}_{\mathcal{T},\mathcal{D}_{\text{train}}(f)}], \mathcal{D}_{\text{eval}}(f)) \quad (2)$$

We denote by $\texttt{adapt}_{\text{mem}}$ the value of this metric over training data ($\mathcal{D}_{\text{eval}} = \mathcal{D}_{\text{train}}$), and by $\texttt{adapt}_{\text{gen}}$ the metric over test data ($\mathcal{D}_{\text{eval}} = \mathcal{D}_{\text{test}}$).

5 Memorizing datasets

Our first experiment investigates the extent to which task adaptation paradigms can memorize different types of tasks. We are interested in memorization because many real NLP tasks involve some degree of memorization. For example, translation builds on memorizing lexical associations between words in various languages, and semantic similarity and paraphrasing require memorizing word meanings and/or groupings of semantically similar words.

Method We use training-set adaptability ($\texttt{adapt}_{\text{mem}}$) as an indicator of a model's memorization ability (Fig. 2). We train on a set of 1000 examples, and plot the value of Eq. (2) on each *atomic* task as models are adapted via fine-tuning or prompt-tuning. This allows us to visualize both the final training-set performance, as well as the time it took to achieve that performance, both of which we use to quantify memorization ability.

Results Figure 2 shows the training curves for fine-tuning (left) and prompt-tuning (right), on different types of tasks. Solid lines show $\texttt{adapt}_{\text{mem}}$, while dashed lines show $\texttt{adapt}_{\text{gen}}$.

Under both adaptation paradigms, we find that lexical tasks are easier to memorize than factual tasks, while random tasks are the hardest to memorize. However, for fine-tuning, we find that models can (eventually) learn to perfectly memorize all types of tasks—even entirely random word associations. However, different types of tasks converge at different rates—lexical tasks converge first, followed by factual tasks, followed by random tasks.

Prompt-tuning, with many fewer parameters than fine-tuning, is much less expressive. As shown in Fig. 2, none of the tasks types converge to 100% accuracy across tasks. Prompt-tuning overall also takes significantly longer to converge; in particular, on random tasks, the finetuned model generally converges at ∼20k updates, while the prompt-tuned model takes over 200k updates to even begin performing nontrivially.

	Atomic	Word-level Comp	Seq Comp
FFT	$46.9_{\pm 4.0}$	$39.5_{\pm 2.1}$	$21.5_{\pm 1.9}$
FPT	$42.6_{\pm 4.3}$	$28.1_{\pm 2.4}$	$11.5_{\pm 1.4}$
32FT	$33.6_{\pm 3.8}$	$22.2_{\pm 1.8}$	$5.7_{\pm 0.9}$
32PT	$32.4_{\pm 3.6}$	$21.7_{\pm 1.7}$	$6.9_{\pm 1.1}$

Table 1: Model (generalization) adaptabilities to atomic, word-level compositional, and sequential compositional tasks, under full fine-tuning (FFT), full prompt-tuning (FPT), 32-shot fine-tuning (32FT) and 32-shot prompt-tuning (32PT). Prompt-tuned models are comparable to fine-tuned models for atomic tasks, but not for compositional tasks. However, this distinction disappears under few-shot learning.

However, despite being much worse at memorization, prompt-tuned models still generalize almost as well as fully fine-tuned models, at least on atomic tasks. This suggests that the inability to memorize arbitrary functions is not necessarily a problem for prompt-tuning in general, and more broadly that overfitting the training set—at least during fine-tuning—may not be necessary for generalization.

In Appendix E, we run a version of this experiment on *permuted* task labels in order to better disentangle the effect of learning novel tasks vs. retrieving existing ones. We find that, for both prompt-tuning and fine-tuning, pre-trained models can more easily adapt to existing relations than to novel (permuted) ones, but they are still *able* to adapt to new tasks, especially compared to non-pre-trained models.

6 Composing tasks

In the previous section, we found that while prompt-tuning cannot memorize arbitrary tasks like fine-tuning, it can still generalize well on simple atomic tasks, almost comparably to fine-tuning. In this section we investigate whether this finding extends to more complex tasks. Specifically, we examine the behavior of prompt-tuned and fine-tuned models when adapted to *compositions* of atomic tasks.

Many prior studies of compositionality focus on *instance-level* compositionality (Lake and Baroni, 2018; Keysers et al., 2020): they test whether models can generalize to new instances by combining information from previously-seen instances *within the same task*. For example, Lake and Baroni (2018) study whether models can learn to *jump left*, after learning to *jump*, *run*, and *run left*. In our work, we instead focus on *task-level* compositionality, studying whether models can adapt to new *tasks* that are compositions of simpler tasks on which they are known to perform well. Thus, while a model exhibiting *compositional generalization* will correctly compose fragments of previously observed training examples, a training procedure exhibiting *compositional adaptability* will perform well on tasks involving compositions of previously learned skills.

Method We study adaptation to complex tasks by relating performance on *atomic* tasks with performance on *depth-2 compositional tasks*. We also study each paradigm under few-shot learning, by creating a random 32-sample subset of each training dataset, and training on that subset. To mitigate the effect of the random seed, we report average performance over 4 different subsets.

What allows models to adapt to these complex tasks? We hypothesize that their adaptability is (in part) compositional—when they can adapt to simple tasks, they can also adapt to compositions of those tasks. For each training paradigm $\mathcal{T}$ and each composition function C, we run linear regression to estimate the Pearson correlation coefficient r^2 between adaptability to a compositional task $C(f_1, \cdots f_n)$,

$$\texttt{adapt}_{\text{gen}}(\mathcal{M}, \mathcal{T}, C(f_1, \cdots, f_n)), \quad (3)$$

and *average* adaptability to the task's atomic components,

$$\frac{1}{n}\sum_{i=1}^{n} \texttt{adapt}_{\text{gen}}(\mathcal{M}, \mathcal{T}, f_i). \quad (4)$$

Figure 3 depicts the procedure graphically.[3]

Can language models learn compositional tasks? The average model adaptability to compositional and atomic tasks, under each training paradigm, is reported in Table 1. We observe that the gap between full-data prompt-tuned models and full-data fine-tuned ones is much larger on compositional tasks than atomic ones. Thus, prompt-tuned models can only generalize comparably to finetuned ones for sufficiently "simple" tasks.

Interestingly, this distinction disappears under few-shot learning. Though both adaptation paradigms generalize much worse in the few-shot

[3]We focus only on compositional functions C which have at least 20 compositional tasks $C(f_1, \cdots, f_n)$ in TASKBENCH500, so that we have at least 20 points to obtain a statistically significant correlation coefficient.

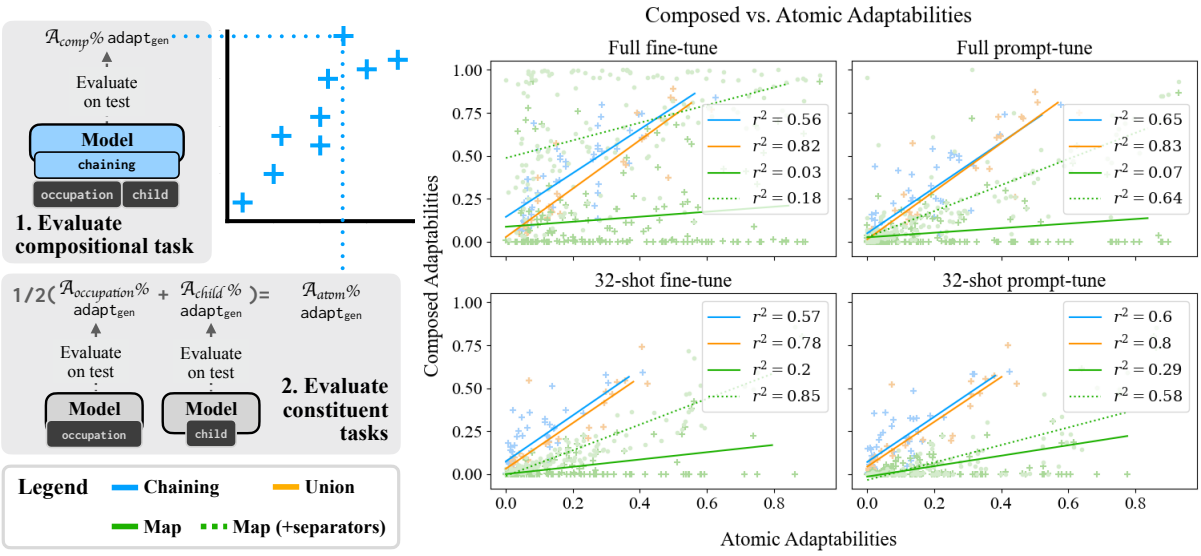

Figure 3: Left: Overview of the composition experiment. We evaluate how well the adaptability on a compositional task can be predicted by the (averaged) adaptabilities of the atomic constituent tasks. Right: Correlation between compositional adaptability vs. averaged atomic adaptabilities, for the chaining, union, and map composition types, under each training paradigm. On word-level chaining and union compositions, compositional adaptability is observed: composed task performance is highly correlated with atomic task performance ($r^2 > 0.5$) under all training paradigms. However, on sequential map compositions, all models perform poorly, and thus non-compositionally. This results from challenges in segmenting input sequences; if token boundaries are explicitly marked (map (+separators)), compositional adaptability is again observed.

setting compared to the full setting, they appear to be comparable to each other in the few-shot setting, even on compositional tasks. This may simply imply that few examples are insufficient to learn the nuances of complex tasks, and that simply learning a few prompt tokens is sufficient to capture what can be learned from the limited data samples.

Do language models adapt compositionally? We visualize each regression model in Fig. 3. Higher r^2 indicates higher correlation between atomic and compositional versions of tasks. Note that *all* model training paradigms demonstrate some degree of word-level compositionality ($r^2 > 0.5$)—when they succeed at word-level compositional tasks (`union`, `chaining`), they succeed at the atomic constituents to those tasks, and vice versa. However, this does not appear to be the case for sequential `map`. In the full-data regime, both fine-tuning and prompt-tuning have near-zero r^2 values. In the few-shot regime, the r^2 value, while nontrivial, is also quite low. Note the slopes of the learned regression lines—the model appears to be unable to learn the sequential versions of tasks, despite succeeding at their atomic versions. To explain this result, we hypothesize that a major obstacle to sequence-level compositional adaptability is *segmentation* of sequences into atomic units. This is especially the case for factual tasks: for example, the sequence `Pauline Payne Whitney Charles Lloyd` could be segmented as `[Pauline Payne Whitney] [Charles Lloyd]` or `[Pauline Payne] [Whitney Charles Lloyd]`, etc. To test whether segmentation is a bottleneck, we train on a version of sequential tasks where we give the language model *explicit* markers of word/entity boundaries (e.g. the language model is given `Pauline Payne Whitney # Charles Lloyd` as input). We found that, with separators, performance on the `map` tasks increases substantially and the model demonstrates compositional adaptability ($r^2 > 0.5$) to these tasks in 3 of the 4 adaptation paradigms. This setting is plotted in Fig. 3 as *Map (+separators)*.

Under this setting, full fine-tuning is the only training paradigm that does not demonstrate compositional adaptability. To better understand this phenomenon, we exclusively plot points from the *Map (+separators)* setting in Appendix Fig. 5. We find that the distribution of points in the full fine-tuning case shows that points tend to fall within the upper-left triangle. This indicates that for a significant number of tasks, *models adapt to their sequential versions despite failing at atomic versions.* In these cases, the model does not simply adapt com-

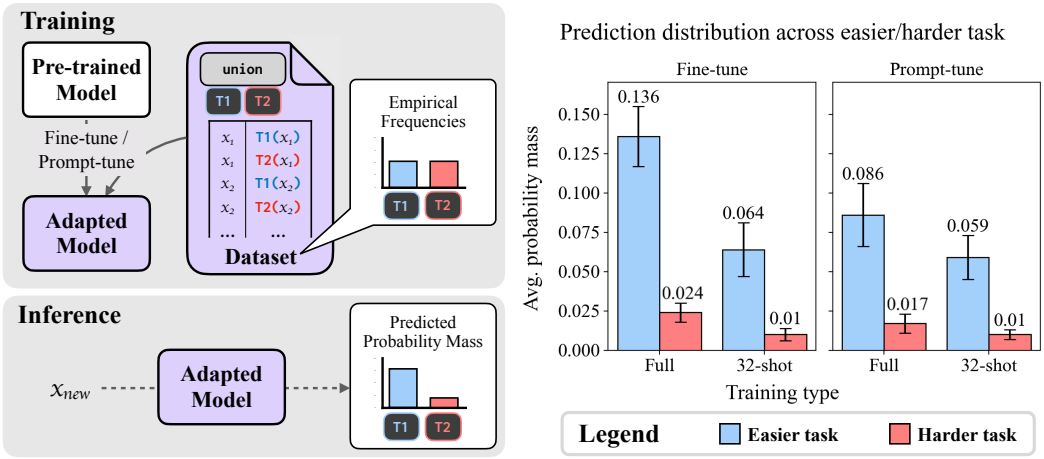

Figure 4: Left: Overview of the task prediction distribution experiments (Section 7). We train a model on a balanced dataset of two tasks, and check whether the prediction distribution over tasks on novel examples matches the (balanced) empirical distribution. Right: Probability mass, across all pairs of tasks, assigned to all answers corresponding to the easier vs. harder task, when trained on a balanced dataset and evaluated novel examples. We report the average across all task pairs and held-out examples, as well as standard errors for each task pair. Note that the model tends to assign more probability to the easier task, despite the task training set being balanced.

positionally, but can take advantage of additional information present in sequences (e.g., seeing more tokens, more examples of the word-level function) to outperform compositional adaptation.

7 Learning new distributions

Previous sections investigated the degree to which models could fit particular tasks using a binary metric that assigned credit to any acceptable answer. Our final set of experiments explores a finer-grained notion of correctness: when there are multiple acceptable answers, as is often the case in real NLP tasks, when does the output *distribution* of a model match the distribution empirically observed during adaptation?

Method We specifically investigate whether models are biased towards predicting "easy" labels, in the sense measured in Section 5. We consider all possible pairs of atomic tasks f_1, f_2 (for which f_1 and f_2 take in overlapping sets of inputs). Let f_e to be the easier task in this pair and f_h be the harder task, relative to a model $\mathcal{M}$ and training paradigm $\mathcal{T}$, in the sense that $\texttt{adapt}_{\text{gen}}(\mathcal{M}, \mathcal{T}, f_e) > \texttt{adapt}_{\text{gen}}(\mathcal{M}, \mathcal{T}, f_h)$. We compose f_e and f_h using *union* to create compositional task $\cup(f_e, f_h)$, and construct the training dataset for this task to be balanced — such that the model sees an equal number of examples of form $(x, f_e(x))$ as $(x, f_h(x))$. Now let $\mathcal{M}_{\cup(f_e,f_h)}$ denote a model adapted to this task. During test-time, we provide $\mathcal{M}_{\cup(f_e,f_h)}$ with novel inputs x' from the domain of both f_e and f_h, and record the average probability mass it assigns to all $y_e^i \in f_e(x')$ versus all $y_h^i \in f_h(x')$.[4] Finally, we average these dataset-wide probabilities over all pairs of tasks, to get an aggregated probability mass assigned to all easier tasks and all harder tasks in a task pair, invariant of the actual underlying task identity. More details on this procedure can be found in Appendix D.

Results Overall, as seen in Fig. 4, across all tasks and training paradigms, the model tends to assign a higher probability to the easier relation. As a concrete example, when trained to predict either antonyms or lexical entailments, the average probability mass placed on the antonyms of a word from the held-out set (easier relation) is 13%, while the average probability mass placed on the entailments of a word (harder relation) is 8%.

Thus, despite having a perfectly balanced fine-tuning set, pretrained models still predict label distributions in a way that align with their inductive biases (measured via the "intrinsic difficulty" of individual labels). This holds for all task adaptation methods, including full fine-tuning, meaning even paradigms and models that *can* fit more complex tasks still have residual biases from pretraining that affect their predictions. This also suggests wider-reaching consequences for model fairness

[4] Note that the model may (and often does) assign mass to answers outside of these sets.

and equity: simply debiasing a fine-tuning dataset is insufficient to overcome biases from pretraining.

8 Conclusion

In this paper, we construct TASKBENCH500, a synthetic task set which serves as a testbed for task adaptability. We focus on three axes of adaptability: ability to memorize, ability to (compositionally) generalize, and ability to fit to novel distributions. We study two adaptation paradigms: fine-tuning and prompt-tuning, finding that: 1. unlike fine-tuning, prompt-tuning cannot memorize completely arbitrary tasks beyond a small number of examples, 2. all adaptation paradigms demonstrate compositional adaptation to word-level compositions, but not sequence-level compositions, and 3. no paradigm is able to perfectly replicate the downstream distribution—all paradigms learn output distributions that align with its inductive biases.

In future work, TASKBENCH500 can be used to study other factors that may affect adaptability, such as length of the prompt in prompt-tuning, similarity between the task distribution and the pretraining distribution, or finer-grained distinctions between tasks (beyond lexical/factual/random, or composition type) that predict task adaptability. TASKBENCH500 can also be used to explore the limitations of *prompt engineering* on a GPT3-scale model. Finally, the current set of tasks and primitives in TASKBENCH500 are by no means complete. Future work can expand on these primitives and study the relationships between the tasks put forth here and real NLP tasks.

Acknowledgements

This work was supported by the MIT–IBM Watson AI lab. Part of the work was done using computing resources provided by a hardware donation from NVIDIA under the NVAIL program, and by the Lincoln Laboratory Supercloud. BZL is supported by an NDSEG Fellowship.

Impact Statement

This paper introduces a new procedure for defining task suites. This procedure is then used to create a 500-task benchmark, which measures the adaptability of pre-trained language models to new tasks. Because the benchmark is created procedurally from databases of words and entities, we anticipate that there should be little to no identifying information or toxic and hateful content. Our datasets should also contain less social bias compared to natural datasets.

However, like with all benchmarks, overfitting to static datasets can inhibit progress in NLP. Moreover, even though this dataset is procedurally generated, we cannot avoid all biases. The resources upon we build our benchmark are themselves biased—for example, lexical databases (like WordNet) are much richer for certain languages (like English) than others, and WikiData currently features many more men than women. Our benchmark currently only features English and Spanish tasks, with a heavy bias towards standard English. This can encourage development of methods that underserve non-standard-English-speaking communities.

We hope to mitigate the aforementioned issues by releasing the code to procedurally generate task suites. We emphasize that the benchmark is dynamic: consisting of not just the static task suite that we are currently releasing, but more importantly the procedure for creating new tasks suites. We encourage future researchers to develop analogous task suites for low-resource languages, non-standard English dialects, and more balanced sets of entities.

References

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Andrea Esuli and Fabrizio Sebastiani. 2006. SENTIWORDNET: A publicly available lexical resource for opinion mining. In *Proceedings of the Fifth International Conference on Language Resources and*

Evaluation (LREC'06), Genoa, Italy. European Language Resources Association (ELRA).

Christiane Fellbaum. 1998. *WordNet: An Electronic Lexical Database*. Bradford Books.

W. N. Francis and H. Kucera. 1979. Brown corpus manual. Technical report, Department of Linguistics, Brown University, Providence, Rhode Island, US.

Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q. Weinberger. 2017. On calibration of modern neural networks. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ICML'17, page 1321–1330. JMLR.org.

Junjie Hu, Sebastian Ruder, Aditya Siddhant, Graham Neubig, Orhan Firat, and Melvin Johnson. 2020. Xtreme: A massively multilingual multi-task benchmark for evaluating cross-lingual generalisation. In *ICML*, pages 4411–4421.

Robin Jia and Percy Liang. 2017. Adversarial examples for evaluating reading comprehension systems. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, EMNLP 2017, Copenhagen, Denmark, September 9-11, 2017*, pages 2021–2031. Association for Computational Linguistics.

Divyanshu Kakwani, Anoop Kunchukuttan, Satish Golla, Gokul N.C., Avik Bhattacharyya, Mitesh M. Khapra, and Pratyush Kumar. 2020. IndicNLPSuite: Monolingual corpora, evaluation benchmarks and pre-trained multilingual language models for Indian languages. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 4948–4961, Online. Association for Computational Linguistics.

Daniel Keysers, Nathanael Schärli, 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 *International Conference on Learning Representations*.

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.

Brenden Lake and Marco Baroni. 2018. Generalization without systematicity: On the compositional skills of sequence-to-sequence recurrent networks. In *International conference on machine learning*, pages 2873–2882. PMLR.

Brian Lester, Rami Al-Rfou, and Noah Constant. 2021. The power of scale for parameter-efficient prompt tuning. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 3045–3059, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Xiang Lisa Li and Percy Liang. 2021. Prefix-tuning: Optimizing continuous prompts for generation. 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 4582–4597, Online. Association for Computational Linguistics.

Nelson F. Liu, Tony Lee, Robin Jia, and Percy Liang. 2021a. Can small and synthetic benchmarks drive modeling innovation? a retrospective study of question answering modeling approaches.

Xiao Liu, Kaixuan Ji, Yicheng Fu, Weng Lam Tam, Zhengxiao Du, Zhilin Yang, and Jie Tang. 2021b. P-tuning v2: Prompt tuning can be comparable to fine-tuning universally across scales and tasks.

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*.

M. Antonia Martí, MarionaTaulé, Lluís Márquez, and Manuel Bertran. 2007. Cess-ece: A multilingual and multilevel annotated corpus.

Rebecca Marvin and Tal Linzen. 2018. Targeted syntactic evaluation of language models. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1192–1202, Brussels, Belgium. Association for Computational Linguistics.

Tom McCoy, Ellie Pavlick, and Tal Linzen. 2019. Right for the wrong reasons: Diagnosing syntactic heuristics in natural language inference. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3428–3448, Florence, Italy. Association for Computational Linguistics.

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.

Sungjoon Park, Jihyung Moon, Sungdong Kim, Won Ik Cho, Ji Yoon Han, Jangwon Park, Chisung Song, Junseong Kim, Youngsook Song, Taehwan Oh, Joohong Lee, Juhyun Oh, Sungwon Lyu, Younghoon Jeong, Inkwon Lee, Sangwoo Seo, Dongjun Lee, Hyunwoo Kim, Myeonghwa Lee, Seongbo Jang, Seungwon Do, Sunkyoung Kim, Kyungtae Lim, Jongwon Lee, Kyumin Park, Jamin

Shin, Seonghyun Kim, Lucy Park, Alice Oh, Jung-Woo Ha, and Kyunghyun Cho. 2021. KLUE: Korean language understanding evaluation. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237, New Orleans, Louisiana. Association for Computational Linguistics.

Fabio Petroni, Tim Rocktäschel, Sebastian Riedel, Patrick Lewis, Anton Bakhtin, Yuxiang Wu, and Alexander Miller. 2019. Language models as knowledge bases? 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 2463–2473, Hong Kong, China. Association for Computational Linguistics.

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.

Kyle Richardson, Hai Hu, Lawrence Moss, and Ashish Sabharwal. 2020. Probing natural language inference models through semantic fragments. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34:8713–8721.

Laura Ruis, Jacob Andreas, Marco Baroni, Diane Bouchacourt, and Brenden M Lake. 2020. A benchmark for systematic generalization in grounded language understanding. In *Advances in Neural Information Processing Systems*, volume 33, pages 19861–19872. Curran Associates, Inc.

David Saxton, Edward Grefenstette, Felix Hill, and Pushmeet Kohli. 2019. Analysing mathematical reasoning abilities of neural models. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net.

Taylor Shin, Yasaman Razeghi, Robert L. Logan IV, Eric Wallace, and Sameer Singh. 2020. AutoPrompt: Eliciting Knowledge from Language Models with Automatically Generated Prompts. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4222–4235, Online. Association for Computational Linguistics.

Daniil Sorokin and Iryna Gurevych. 2018. Modeling semantics with gated graph neural networks for knowledge base question answering. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 3306–3317, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

Denny Vrandečić and Markus Krötzsch. 2014. Wikidata: A free collaborative knowledgebase. *Commun. ACM*, 57(10):78–85.

Alex Wang, Yada Pruksachatkun, Nikita Nangia, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. 2019. SuperGLUE: A stickier benchmark for general-purpose language understanding systems. In *NeurIPS*.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 2018. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 353–355, Brussels, Belgium. Association for Computational Linguistics.

Jason Weston, Antoine Bordes, Sumit Chopra, and Tomás Mikolov. 2016. Towards ai-complete question answering: A set of prerequisite toy tasks. In *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*.

Liang Xu, Hai Hu, Xuanwei Zhang, Lu Li, Chenjie Cao, Yudong Li, Yechen Xu, Kai Sun, Dian Yu, Cong Yu, Yin Tian, Qianqian Dong, Weitang Liu, Bo Shi, Yiming Cui, Junyi Li, Jun Zeng, Rongzhao Wang, Weijian Xie, Yanting Li, Yina Patterson, Zuoyu Tian, Yiwen Zhang, He Zhou, Shaoweihua Liu, Zhe Zhao, Qipeng Zhao, Cong Yue, Xinrui Zhang, Zhengliang Yang, Kyle Richardson, and Zhenzhong Lan. 2020. CLUE: A Chinese language understanding evaluation benchmark. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 4762–4772, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. 2017. Understanding deep learning requires rethinking generalization. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net.

A More details on TASKBENCH500 creation procedure

A.1 Task creation details

For atomic lexical tasks, we take a subset of relations specified in either Wordnet (Fellbaum, 1998) or SentiWordNet (Esuli and Sebastiani, 2006). For atomic factual tasks, we take a subset of tasks from Wikidata (Vrandečić and Krötzsch, 2014). We also have 3 broad categories of composition functions: set operations, logical operations, and sequential operations. The full list of atomic tasks can be found in Table 4 and the list of composition functions can be found in Table 5.

We enumerate all possible depth-2 word level compositions of each task, and the sequential versions of them (i.e. if the task is a relation, inserting it into a `map`, or if the task is a predicate, inserting it into a `filter`), up to 500 tasks. We also apply some basic heuristics to filter identical tasks: for example, we filter symmetric relations, e.g. `union(B,A)` is identical to `union(A,B)`, or avoid the use of logical operations alongside set operations, e.g. `lor(in(x,A), in(x,B))` is identical to `in(x,union(A,B))`. Our full list of tasks can be found in Tables 4, 6, 7 and 8.

Sequential compositions Sequential composition functions convert word-wise tasks to sequence-level tasks. We specifically consider only two sequential functions: `map` and `filter`. Note that compositions of multiple `maps` or multiple `filters` can instead be expressed as compositions of multiple word-level functions. For example,

$$\texttt{map}\{\lambda\texttt{x.occupation(x)}\}(\texttt{map}\{\lambda\texttt{x.father(x)}\}(\texttt{S}))$$

(for an input sequence `S`) is equivalent to

$$\texttt{map}\{\lambda\texttt{x.occupation(father(x))}\}(\texttt{S})$$

Specifically, we define the following top-level sequential operator

$$\begin{aligned}&\texttt{map-filter}\{f_M, f_F\}\\&\quad=\texttt{map}\{f_M\}(\texttt{filter}\{f_F\})\end{aligned} \tag{5}$$

where f_M is a word-wise relation and f_F is a word-wise predicate. All recursively-defined sequential operators follow this form. The following are the recursive rules for mapping nested maps and filters into a function of this form: in the base cases,

$$\begin{aligned}\texttt{map}\{f_M\} &= \texttt{map-filter}\{f_M, \lambda\texttt{x.true}\}\\ \texttt{filter}\{f_F\} &= \texttt{map-filter}\{\lambda\texttt{x.x}, f_F\};\end{aligned} \tag{6}$$

in the recursive cases,

$$\begin{aligned}&\texttt{map}\{f'_M\}(\texttt{map-filter}\{f_M, f_F\})\\&\quad=\texttt{map-filter}\{f'_M(f_M), f_F\}\\&\texttt{filter}\{f'_F\}(\texttt{map-filter}\{f_M, f_F\})\\&\quad=\texttt{map-filter}\{f_M, f_F \wedge f'_F(f_M)\}.\end{aligned} \tag{7}$$

A.2 Dataset creation details

Note that many tasks created through composition will be degenerate or identical to other tasks, even with our heuristic filters. We do a preliminary filter for degenerate tasks by removing tasks for which we have less than 100 samples. We also manually inspect all depth-2 word-level lexical compositions to ensure they are nontrivial and unique.

Word-level lexical tasks For English lexical tasks, we use words that appeared more than 5 times in the Brown corpus (Francis and Kucera, 1979) as our inputs x. For Spanish lexical tasks, we in use words that appeared at least once in the CESS Spanish Treebank (Martí et al., 2007) as our inputs. This results in a a total of 9143 English words and 5298 Spanish words. We then construct outputs for each input word using either WordNet or SentiWordNet. From each task, we filter out samples for which the relations map to an empty set—thus, for a task like `intersection(synonym(x), antonym(x))`, most samples will be filtered out as the set of synonyms are usually disjoint from the set of antonyms. (This task ends up getting filtered out entirely, as the final number of samples is under 100.)

Word-level factual tasks We use a dump of Wikidata from 2017, taken from Sorokin and Gurevych (2018).[5] We convert each word-level factual task into SPARQL queries which returns a set of input-output data pairs from Wikidata.

For factual relations R, we create two queries: a *sample* query which gives us a set of entities that participate in the relation, from which the inputs x are derived, and a *function* query that maps specific inputs x to its set of output entities $R(x)$. For factual predicates P, we create three queries: a *positive sample* query which gives samples x for which $P(x) = \texttt{true}$, a *negative sample* query which gives samples x for which $P(x) = \texttt{false}$,

[5] https://public.ukp.informatik.tu-darmstadt.de/wikidata-dump/wikidata-virtuoso-dump-2017.zip

Task (T)	SPARQL fragment (sparql(T, y))
A(x)	?x A ?y .
union(T1(x),T2(x))	{ sparql(T1(x), y) } UNION { sparql(T2(x), y) }
intersection(T1(x),T2(x))	sparql(T1(x), y) sparql(T2(x), y)
lor(T1(x),T2(x))	BIND(y1 || y2 AS y) sparql(T1(x), y1) sparql(T2(x), y2)
land(T1(x),T2(x))	BIND(y1 && y2 AS y) sparql(T1(x), y1) sparql(T2(x), y2)

Table 2: Rules for mapping word-level factual tasks to SPARQL conditional statements. Blue substrings represent recursive calls to this set of rules, which are to be replaced with their output SPARQL fragments. Note the second argument to the `sparql` function represents the variable name to output to.

and a *function* query that maps specific inputs x to its output boolean value $P(x)$.

The SPARQL query is generated recursively given the specification of the task. We define a function `task2sparql(T(x),y)` which converts tasks `T(x)` to SPARQL fragments (where the second argument to the function is the variable name we define for the output). We then convert the output of this function into a well-formed query using:

```
SELECT ?x
WHERE <task2sparql(T(x),y)>
```

for sample queries and

```
SELECT ?y
WHERE <task2sparql(T(x),y)>
```

for function queries. Note for function queries that the input `x` is provided to us (and is not a variable).

The general rules specifying the `task2sparql` function are given in Table 2.

Sequential tasks In practice, naively concatenating outputs from a random word sampler to create sequences will return degenerate or trivial sequences for many functions (for example, `map{`λ`x. child(x)}` is not meaningful for sequences consisting of words that don't refer to humans). Thus, we define a sequence sampler in Algorithm 1 that takes in a sequential function (given in the form from Eq. (5)), an input length n and an output length $m \leq n$, which will always sample sequences with length n such that the output, when the function is applied to the sequence, is of length m.

At a high level, this algorithm samples n input words which are in the domain of the map relation, and for which the filter predicate returns `true`, and $m - n$ input words for which the filter predicate returns `false`, then permutes and concatenates them.

Algorithm 1: Algorithm for sampling meaningful input sequences for sequential tasks.

function *seq_sampler*(`map-filter`(f_M, f_F), n, m)**:**
  $seq \leftarrow$ "";
  for $i = 1 \cdots n$ **do**
    $word \sim$ `Unif`(`domain`(f_M) $\cap \{x : f_F(x) =$ `true`$\}$);
    $seq \leftarrow seq + word$
  end
  for $j = n \cdots m$ **do**
    $word \sim$ `Unif`($\{x : f_F(x) =$ `false`$\}$);
    $seq \leftarrow seq + word$
  end
  $seq \leftarrow$ `permute-words`(seq)

B Experimental Setup Details

Hyperparameters We adapt a pre-trained T5-base model (24-layer, 220M parameters) to our tasks. We use an AdamW optimizer with a learning rate of 1.0 for all prompt-tuning experiments, and learning rate of 1e-3 for all fine-tuning experiments. We use batch sizes of 64 for word-level tasks, and 32 for sequential tasks. We run all experiments up to 100 epochs, and run 3–4 trials for each few-shot experiment to estimate average performance over possible choices of few-shot training samples. These hyperparameters were chosen by trial and error on top of default configurations.

Infrastructure and Reproducibility For each task, we adapt our model using a single 32GB NVIDIA V100 GPU, or a single 40GB NVIDIA A100 GPU. Training time varies by training dataset size and maximum number of epochs, but on average (using the hyperparameters specified above) is less than a few hours per task. Prompt-tuning is also more efficient than fine-tuning, updating the parameters of only 100 prompt tokens vs. the full 220M parameters in the model.

Evaluation of Sequential Tasks When evaluating accuracies of sequential tasks (Eq. (1)), note that we must align words in the generated sequence y'_i with words in the ground-truth sequence y_i. However, this can be nontrivial, especially under the setting where word and entity boundaries are

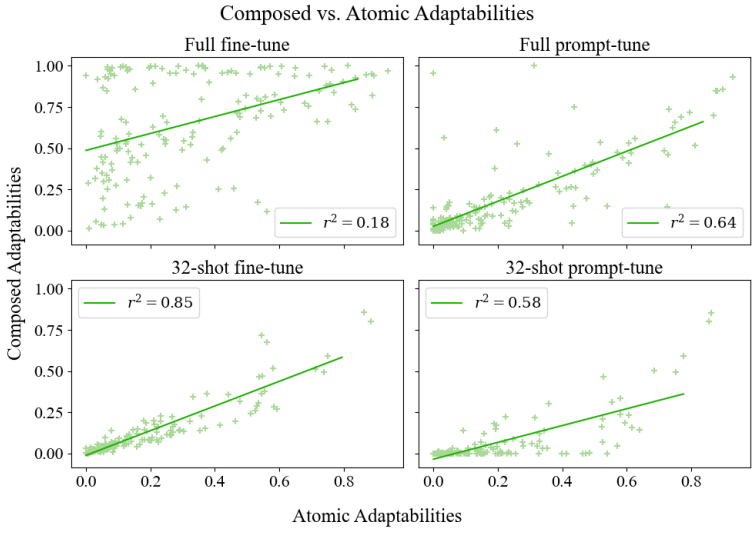

Figure 5: Compositionality of `map` function, when token separators are explicitly provided in the input and output. All adaptation paradigms demonstrate compositionality except for full fine-tuning, where there seems to be a large proportion of tasks for which the model can adapt to sequentially but not atomically.

not explicitly generated by the model. We cannot rely on whitespaces to segment words as a single word can span multiple white-spaces; for example, an entity `Will Smith` constitutes a single word. Instead, given a ground-truth sequence of n words (note ground-truth segmentations are present in the dataset), we optimize over all possible length-n segmentations of the generated sequence.

C Compositionality Experiment: Additional Results

Additional results for the compositionality experiment, including all composition functions, and the formula for the best-fit regression line in each case, are reported in Table 3. Furthermore, the `map` task with explicit segmentation (+separators) is plotted in isolation in Fig. 5.

D Prediction distribution experiment: Additional details

We adapt the model to the task $\cup(f_e, f_h)$, constructing the training dataset for $\cup(f_e, f_h)$ to be balanced — such that the model sees an equal number of examples of form $(x, f_e(x))$ as $(x, f_h(x))$.

Let $\mathcal{M}_{\cup(f_e,f_h)}$ denote a model adapted to this task. Note that the domains of either function are not always identical, for example the set of entities in the domain of `political-party-of(x)` (mostly politicians) is different from the set of entities in the domain of `position-played-on-sports-team(x)` (mostly athletes). We create a balanced training set by first taking all items in the intersection of both domains, then sampling an equal number number of items in either domain. Furthermore, to minimize the effect of the order seen during training, we shuffle the entire dataset after creating all example-label pairs.

During test-time, we give $\mathcal{M}_{\cup(f_e,f_h)}$ a novel input x' and record the average probability mass it assigned to all $y_e^i \in f_e(x')$ vs. all $y_h^i \in f_h(x')$. Note we evaluate only on inputs x' which are in the domain of both f_e and f_h. Under the rare scenario that a prediction is in *both* target tasks for a particular word (i.e. y is in both $f_e(x')$ and $f_h(x')$), we count that towards both tasks, and increment the probability mass on both tasks by the probability the model assigned to y.

Instead of averaging across outputs in either set $f_e(x'), f_h(x')$, we also looked at the probabilities assigned to *highest*-scoring predictions from each set. The overall trends were similar: the model tends to assign greater mass to the highest-scoring prediction from the easier task compared to highest-scoring prediction from the harder task.

E Permuting task labels: disentangling effect of "learning" vs. "retrieval"

We hypothesize two ways that pre-trained models might adapt to new tasks: (1) through learning the underlying rules and patterns governing the task, or (2) through learning how to "retrieve" the correct label from memorized pre-training data. These hy-

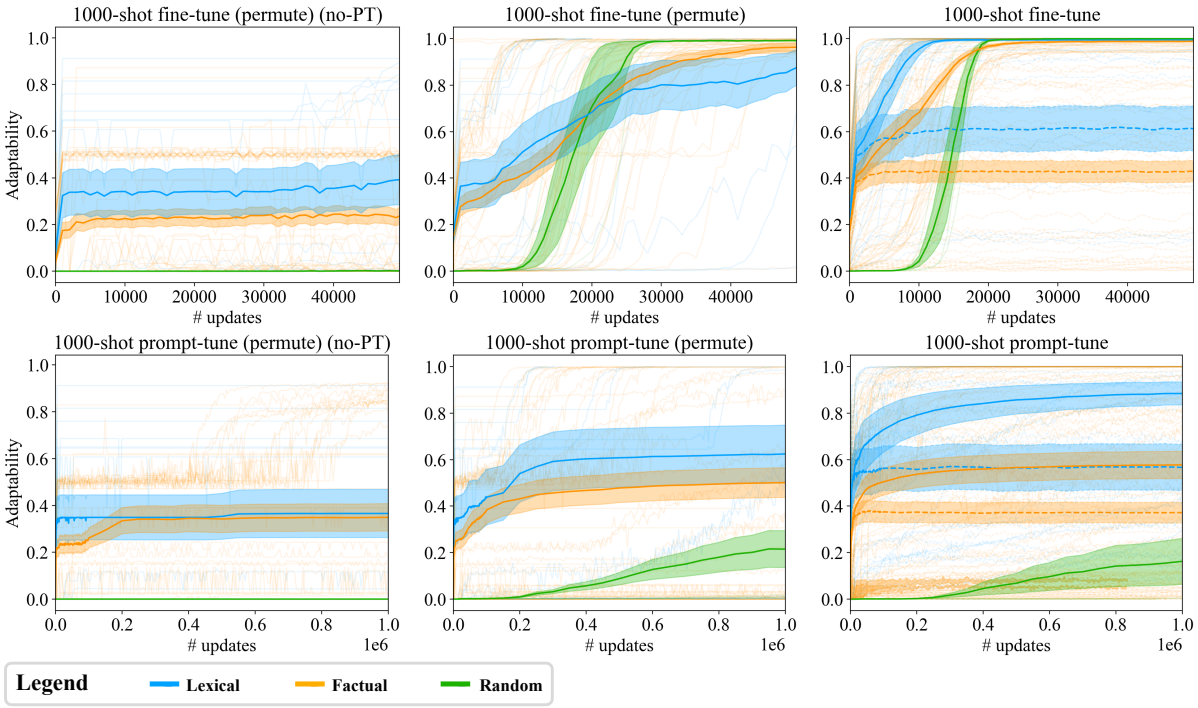

Figure 6: Memorization experiments on permuted vs. non-permuted versions of tasks, using pre-trained vs. non-pretrained models. Left figure shows an averaged memorization curve for a non-pretrained model on permuted tasks. Middle figure shows a pre-trained model on permuted tasks. Right figure shows a pre-trained model on non-permuted tasks. Pre-training enables models to adapt to novel tasks, but adapting to existing, non-permuted tasks is easier than adapting to novel, permuted tasks.

potheses, respectively, suggest two different roles for pre-training: (1) providing a "generally good" initialization from which many different tasks can be learned, or (2) imbuing the LM with memorized knowledge that can later be retrieved.

To determine which effect is at play (for which types of tasks), for each atomic task, we permute the labels associated with each input, then run each adaptation paradigm on the permuted version of the task. Notably, permuted labels differ from `random` tasks as the input and label distributions are restricted to be identical to original task. Because the model would be unable to generalize to permuted labels, we only look at memorization ability. The setting is similar to Section 5. We compare the rate of adaptation for (A) a non-pretrained model to a permuted task, (B) a pre-trained model to a permuted task, and (C) a pre-trained model to non-permuted task. If a pre-trained model is better able to adapt to a task than the non-pretrained model (B $>$ A) , this indicates that pre-training helps models learn new tasks on the fly, supporting hypothesis 1. If a pre-trained model can better adapt to a non-permuted task than it can to a permuted task (C $>$ B), this indicates that adaptation requires something learned during pre-training, supporting hypothesis 2.

Results are shown in Fig. 6 (which, from left to right, shows settings A-C respectively). We find that for fine-tuning and prompt-tuning, both hypotheses are supported. For both lexical and factual tasks, pre-trained models can memorize novel word relations faster than non-pre-trained models. However, pre-trained models can still adapt to non-permuted tasks faster than permuted ones. Furthermore, note that for fine-tuning, the order of convergence of the three task types is reversed when going from permuted tasks to non-permuted tasks. In particular, random relations are easier to learn than permuted lexical or factual tasks. This suggests that *models can more easily to adapt to random labels than labels that are known to be false*.

Function type	Training type	Avg. adaptability	Optimal formula	r^2 value
Chaining $f_2(f_1)$	Full Fine-tuning	$37.43_{\pm 3.18}$	$1.27x + 0.14$	0.56
	Full Prompt-tuning	$22.37_{\pm 3.03}$	$1.32x + 0.05$	0.65
	32-shot Fine-tuning	$18.59_{\pm 2.21}$	$1.34x + 0.07$	0.57
	32-shot Prompt-tuning	$18.19_{\pm 2.21}$	$1.32x + 0.07$	0.6
Union $f_2 \cup f_1$	Full Fine-tuning	$31.18_{\pm 2.02}$	$1.24x + 0.02$	0.73
	Full Prompt-tuning	$25.05_{\pm 2.11}$	$1.4x - 0.01$	0.83
	32-shot Fine-tuning	$17.28_{\pm 1.52}$	$1.37x + 0.02$	0.8
	32-shot Prompt-tuning	$18.43_{\pm 1.55}$	$1.35x + 0.02$	0.8
Intersection $f_2 \cap f_1$	Full Fine-tuning	$43.31_{\pm 22.42}$	$2.25x - 0.12$*	0.97*
	Full Prompt-tuning	$16.68_{\pm 8.78}$	$1.64x - 0.04$*	0.98*
	32-shot Fine-tuning	$22.77_{\pm 17.03}$	$5.93x - 0.12$*	0.91*
	32-shot Prompt-tuning	$25.91_{\pm 19.38}$	$6.81x - 0.12$*	0.94*
Logical And $f_1 \wedge f_2$	Full Fine-tuning	$78.39_{\pm 2.53}$	$2.15x - 0.85$*	0.8*
	Full Prompt-tuning	$79.25_{\pm 2.57}$	$1.27x - 0.18$*	0.58*
	32-shot Fine-tuning	$66.49_{\pm 2.55}$	$4.75x - 2.13$*	0.88*
	32-shot Prompt-tuning	$55.86_{\pm 1.22}$	$0.48x + 0.3$*	0.05*
Logical Or $f_1 \vee f_2$	Full Fine-tuning	$72.41_{\pm 1.97}$	$1.39x - 0.37$*	0.54*
	Full Prompt-tuning	$74.71_{\pm 2.01}$	$1.15x - 0.18$*	0.48*
	32-shot Fine-tuning	$58.04_{\pm 1.11}$	$1.52x - 0.35$*	0.63*
	32-shot Prompt-tuning	$53.91_{\pm 0.48}$	$0.8x + 0.1$*	0.33*
Map `map{`$\lambda x.f_M(x)$`}`	Full Fine-tuning	$13.44_{\pm 1.73}$	$0.15x + 0.09$	0.03
	Full Prompt-tuning	$5.39_{\pm 0.93}$	$0.13x + 0.03$	0.07
	32-shot Fine-tuning	$3.59_{\pm 0.70}$	$0.21x + 0.0$	0.2
	32-shot Prompt-tuning	$3.77_{\pm 0.85}$	$0.3x - 0.01$	0.29
Map (+separators) `map{`$\lambda x.f_M(x)$`}`	Full Fine-tuning	$67.40_{\pm 2.51}$	$0.49x + 0.52$	0.17
	Full Prompt-tuning	$18.02_{\pm 1.96}$	$0.83x + 0.02$	0.64
	32-shot Fine-tuning	$10.66_{\pm 1.34}$	$0.79x - 0.01$	0.86
	32-shot Prompt-tuning	$5.22_{\pm 1.14}$	$0.57x - 0.04$	0.64
Filter `filter{`$\lambda x.f_F(x)$`}`	Full Fine-tuning	$82.08_{\pm 5.92}$	$1.59x - 0.58$*	0.95*
	Full Prompt-tuning	$78.58_{\pm 5.43}$	$1.38x - 0.43$*	0.95*
	32-shot Fine-tuning	$38.39_{\pm 3.27}$	$0.81x - 0.24$*	0.87*
	32-shot Prompt-tuning	$51.58_{\pm 4.99}$	$1.19x - 0.43$*	0.87*

Table 3: We study the correlation between the atomic word-level functions and their compositions, under various training paradigms. We train a linear regressor to predict a model's generalization adaptability on a composite function based on its adaptabilities on the atomic constituents. Finally, we report the average generalization adaptability of composite tasks, for each training paradigm, under each type of composition.
* indicates composition function has less than 20 tasks, thus reported numbers may not be significant.

Category	Predicates	Relations	
Lexical	is-POS-noun[eng] is-POS-verb[eng] is-POS-adjective[eng] is-POS-adverb[eng] is-sentiment-positive[eng] is-sentiment-negative[eng] is-sentiment-neutral[eng]	synonyms[eng] antonyms[eng] hyponyms[eng] entailments[eng] translate[eng->spa]	synonyms[spa] antonyms[spa] hyponyms[spa] entailments[spa] translate[spa->eng]
Factual	is-instance-human is-instance-film is-instance-book is-instance-city is-instance-taxon is-occupation-actor is-occupation-politician is-occupation-writer is-occupation-journalist is-occupation-teacher is-occupation-composer is-birthplace-london is-birthplace-nyc is-birthplace-la is-birthplace-buenosaires	child child[inv] continent country of citizenship country of origin country creator creator[inv] developer diplomatic relation father father[inv] genre has part head of state head of state[inv] influenced by languages spoken written or signed location	location[inv] manufacturer member of political party member of sports team mother mother[inv] named after native language occupation official language original language of film or TV show owned by performer place of birth place of death position held position played on team record label sex or gender
Random		random-seed0[eng] random-seed1[eng]	random-seed2[eng] random-seed3[eng]

Table 4: Full list of atomic tasks in TASKBENCH500. The content inside brackets specifies task input and output languages (eng for English and spa for Spanish). {inv} indicates the task is inverted, e.g. creator takes creations as input and returns their creators, while creator{inv} takes creators as input and returns their creations.

Category	Function	Example Tasks	Example Data
Chaining	chain	mother(head of state)	Russia → {Maria Ivanovna Putina}
Set Operations	union	union(mother, father)	Elizabeth I of England → {Anne Boleyn, Henry VIII of England}
	intersection	intersection(entailments[eng], synonyms[eng])	live → {be, exist}
Logical Operations	land	land(is-occupation-actor, is-birthplace-nyc)	Anne Hathaway → true Brad Pitt → false Franklin Delano Roosevelt → false
	lor	lor(is-birthplace-london, is-birthplace-nyc)	Franklin Delano Roosevelt → true David Bekham → true Mao Zedong → false
Sequential Operations	map	map{λx. synonyms[eng]}(S)	criticality pillow delinquent culture eternity cling sane sentry → {· · · , criticalness rest neglectful acculturation timelessness cohere reasonable spotter, · · · }
	filter	λx. filter{is-POS-noun[eng]}(S)	expect inexpensive direct bones sullen breed switching eight → {bones breed switching eight}

Table 5: Full list of composition functions used in TASKBENCH500, with examples.

antonyms[eng](entailments[eng])	entailments[spa](antonyms[spa])	influenced by(creator)	mother(creator)	place of birth(influenced by)
antonyms[eng](hyponyms[eng])	entailments[spa](hyponyms[spa])	influenced by(father)	mother(father)	place of birth(named after)
antonyms[eng](translate[spa->eng])	father(creator)	influenced by(influenced by)	mother(head of state)	place of death(influenced by)
antonyms[spa](hyponyms[spa])	father(father)	influenced by(performer)	mother(influenced by)	place of death(named after)
child(influenced by)	father(head of state)	languages spoken written or signed(child)	mother(mother)	position held(influenced by)
child(named after)	father(mother)	languages spoken written or signed(influenced by)	mother(named after)	position held(mother)
child(owned by)	father(named after)	languages spoken written or signed(named after)	mother(owned by)	position played on team(father)
country of citizenship(child)	hyponyms[eng](antonyms[eng])	member of political party(father)	mother(performer)	position played on team(named after)
country of citizenship(father)	hyponyms[eng](entailments[eng])	member of political party(influenced by)	named after(child)	record label(child)
country of citizenship(mother)	hyponyms[eng](translate[spa->eng])	member of political party(mother)	named after(developer)	record label(father)
entailments[eng](antonyms[eng])	hyponyms[spa](antonyms[spa])	member of political party(named after)	named after(influenced by)	record label(influenced by)
entailments[eng](hyponyms[eng])	hyponyms[spa](entailments[spa])	member of sports team(child)	occupation(influenced by)	record label(mother)
entailments[eng](translate[spa->eng])	influenced by(child)	member of sports team(father)	occupation(named after)	translate[eng->spa](antonyms[eng])
entailments[spa](antonyms[spa])	influenced by(creator)			

union(antonyms[eng], entailments[eng])	union(father, mother)	union(place of death, position held)
union(antonyms[eng], hyponyms[eng])	union(hyponyms[eng], synonyms[eng])	union(place of death, position played on team)
union(antonyms[eng], synonyms[eng])	union(hyponyms[spa], synonyms[spa])	union(place of death, record label)
union(antonyms[spa], entailments[spa])	union(languages spoken written or signed, member of political party)	union(random-seed0[eng], antonyms[eng])
union(antonyms[spa], hyponyms[spa])	union(languages spoken written or signed, mother)	union(random-seed0[eng], entailments[eng])
union(antonyms[spa], synonyms[spa])	union(languages spoken written or signed, occupation)	union(random-seed0[eng], hyponyms[eng])
union(child, father)	union(languages spoken written or signed, place of birth)	union(random-seed0[eng], synonyms[eng])
union(child, mother)	union(languages spoken written or signed, position held)	union(random-seed1[eng], antonyms[eng])
union(country of citizenship, languages spoken written or signed)	union(languages spoken written or signed, position played on team)	union(random-seed1[eng], entailments[eng])
union(country of citizenship, named after)	union(languages spoken written or signed, record label)	union(random-seed1[eng], hyponyms[eng])
union(country of citizenship, position held)	union(member of political party, mother)	union(random-seed1[eng], synonyms[eng])
union(country of citizenship, position played on team)	union(member of political party, place of birth)	union(random-seed2[eng], antonyms[eng])
union(creator, father)	union(member of political party, record label)	union(random-seed2[eng], entailments[eng])
union(creator, mother)	union(member of sports team, mother)	union(random-seed2[eng], hyponyms[eng])
union(entailments[eng], hyponyms[eng])	union(member of sports team, place of death)	union(random-seed2[eng], synonyms[eng])
union(entailments[eng], synonyms[eng])	union(occupation, place of death)	union(random-seed3[eng], antonyms[eng])
union(entailments[spa], hyponyms[spa])	union(place of birth, position held)	union(random-seed3[eng], entailments[eng])
union(entailments[spa], synonyms[spa])	union(place of birth, position played on team)	union(random-seed3[eng], hyponyms[eng])
union(father, influenced by)	union(place of birth, record label)	union(random-seed3[eng], synonyms[eng])

intersection(entailments[eng], synonyms[eng])	intersection(hyponyms[eng], synonyms[eng])	intersection(hyponyms[spa], synonyms[spa])

land(is-occupation-actor, is-birthplace-buenosaires)	land(is-occupation-actor, is-birthplace-london)	land(is-occupation-politician, is-birthplace-london)
land(is-occupation-actor, is-birthplace-la)	land(is-occupation-actor, is-birthplace-nyc)	land(is-occupation-politician, is-birthplace-nyc)

lor(is-birthplace-buenosaires, is-occupation-journalist)	lor(is-birthplace-la, is-birthplace-london)	lor(is-birthplace-nyc, is-birthplace-london)	lor(is-occupation-actor, is-birthplace-la)
lor(is-birthplace-buenosaires, is-occupation-politician)	lor(is-birthplace-london, is-occupation-teacher)	lor(is-birthplace-nyc, is-occupation-actor)	lor(is-occupation-actor, is-birthplace-nyc)
lor(is-birthplace-buenosaires, is-occupation-teacher)	lor(is-birthplace-nyc, is-birthplace-buenosaires)	lor(is-birthplace-nyc, is-occupation-politician)	lor(is-occupation-journalist, is-birthplace-buenosaires)
lor(is-birthplace-la, is-birthplace-buenosaires)	lor(is-birthplace-nyc, is-birthplace-la)	lor(is-occupation-actor, is-birthplace-buenosaires)	lor(is-occupation-politician, is-birthplace-buenosaires)
lor(is-birthplace-la, is-birthplace-london)			

Table 6: Full list of word-level compositional tasks in TASKBENCH500, organized by composition type.

map{λx. antonyms[eng](entailments[eng](x))}	map{λx. languages spoken written or signed(child(x))}	map{λx. translate[eng->spa](x)}
map{λx. antonyms[eng](hyponyms[eng](x))}	map{λx. languages spoken written or signed(influenced by(x))}	map{λx. translate[spa->eng](x)}
map{λx. antonyms[eng](translate[spa->eng](x))}	map{λx. languages spoken written or signed(named after(x))}	map{λx. union(antonyms[eng](x), entailments[eng](x))}
map{λx. antonyms[eng](x)}	map{λx. languages spoken written or signed(x)}	map{λx. union(antonyms[eng](x), hyponyms[eng](x))}
map{λx. antonyms[spa](entailments[spa](x))}	map{λx. location(x)}	map{λx. union(antonyms[eng](x), synonyms[eng](x))}
map{λx. antonyms[spa](hyponyms[spa](x))}	map{λx. location[inv](x)}	map{λx. union(antonyms[spa](x), entailments[spa](x))}
map{λx. antonyms[spa](x)}	map{λx. manufacturer(x)}	map{λx. union(antonyms[spa](x), hyponyms[spa](x))}
map{λx. child(influenced by(x))}	map{λx. member of political party(father(x))}	map{λx. union(antonyms[spa](x), synonyms[spa](x))}
map{λx. child(named after(x))}	map{λx. member of political party(influenced by(x))}	map{λx. union(child(x), father(x))}
map{λx. child(owned by(x))}	map{λx. member of political party(mother(x))}	map{λx. union(child(x), mother(x))}
map{λx. child[inv](x)}	map{λx. member of political party(named after(x))}	map{λx. union(child(x), named after(x))}
map{λx. continent(x)}	map{λx. member of political party(x)}	map{λx. union(country of citizenship(x), languages spoken written or signed(x))}
map{λx. country of citizenship(child(x))}	map{λx. member of sports team(child(x))}	map{λx. union(country of citizenship(x), named after(x))}
map{λx. country of citizenship(father(x))}	map{λx. member of sports team(father(x))}	map{λx. union(country of citizenship(x), position held(x))}
map{λx. country of citizenship(mother(x))}	map{λx. member of sports team(influenced by(x))}	map{λx. union(country of citizenship(x), position played on team(x))}
map{λx. country of citizenship(x)}	map{λx. member of sports team(x)}	map{λx. union(creator(x), father(x))}
map{λx. country of origin(x)}	map{λx. mother(creator(x))}	map{λx. union(creator(x), location(x))}
map{λx. country(x)}	map{λx. mother(father(x))}	map{λx. union(creator(x), mother(x))}
map{λx. creator(x)}	map{λx. mother(head of state(x))}	map{λx. union(entailments[eng](x), hyponyms[eng](x))(S)mapchild(x)}
map{λx. creator[inv](x)}	map{λx. mother(influenced by(x))}	map{λx. union(entailments[eng](x), synonyms[eng](x))}
map{λx. developer(x)}	map{λx. mother(mother(x))}	map{λx. union(entailments[spa](x), hyponyms[spa](x))}
map{λx. diplomatic relation(x)}	map{λx. mother(named after(x))}	map{λx. union(entailments[spa](x), synonyms[spa](x))}
map{λx. entailments[eng](antonyms[eng](x))}	map{λx. mother(owned by(x))}	map{λx. union(father(x), influenced by(x))}
map{λx. entailments[eng](hyponyms[eng](x))}	map{λx. mother(performer(x))}	map{λx. union(father(x), mother(x))}
map{λx. entailments[eng](translate[spa->eng](x))}	map{λx. mother(x)}	map{λx. union(father(x), named after(x))}
map{λx. entailments[eng](x)}	map{λx. mother[inv](x)}	map{λx. union(hyponyms[eng](x), synonyms[eng](x))}
map{λx. entailments[spa](antonyms[spa](x))}	map{λx. named after(child(x))}	map{λx. union(hyponyms[spa](x), synonyms[spa](x))}
map{λx. entailments[spa](hyponyms[spa](x))}	map{λx. named after(creator(x))}	map{λx. union(influenced by(x), mother(x))}
map{λx. entailments[spa](x)}	map{λx. named after(developer(x))}	map{λx. union(influenced by(x), named after(x))}
map{λx. father(creator(x))}	map{λx. named after(father(x))}	map{λx. union(languages spoken written or signed(x), member of political party(x))}
map{λx. father(father(x))}	map{λx. named after(influenced by(x))}	map{λx. union(languages spoken written or signed(x), member of sports team(x))}
map{λx. father(head of state(x))}	map{λx. named after(x)}	map{λx. union(languages spoken written or signed(x), mother(x))}
map{λx. father(mother(x))}	map{λx. native language(x)}	map{λx. union(languages spoken written or signed(x), occupation(x))}
map{λx. father(named after(x))}	map{λx. occupation(influenced by(x))}	map{λx. union(languages spoken written or signed(x), place of birth(x))}
map{λx. father(x)}	map{λx. occupation(named after(x))}	map{λx. union(languages spoken written or signed(x), position held(x))}
map{λx. father[inv](x)}	map{λx. occupation(x)}	map{λx. union(languages spoken written or signed(x), position played on team(x))}
map{λx. genre(x)}	map{λx. official language(x)}	map{λx. union(languages spoken written or signed(x), record label(x))}
map{λx. has part(x)}	map{λx. original language of film or TV show(x)}	map{λx. union(member of political party(x), mother(x))}
map{λx. head of state(x)}	map{λx. owned by(x)}	map{λx. union(member of political party(x), named after(x))}
map{λx. head of state[inv](x)}	map{λx. performer(x)}	map{λx. union(member of political party(x), place of birth(x))}
map{λx. hyponyms[eng](antonyms[eng](x))}	map{λx. place of birth(influenced by(x))}	map{λx. union(member of political party(x), record label(x))}
map{λx. hyponyms[eng](entailments[eng](x))}	map{λx. place of birth(named after(x))}	map{λx. union(member of sports team(x), mother(x))}
map{λx. hyponyms[eng](translate[spa->eng](x))}	map{λx. place of birth(x)}	map{λx. union(member of sports team(x), place of death(x))}
map{λx. hyponyms[eng](x)}	map{λx. place of death(influenced by(x))}	map{λx. union(member of sports team(x), record label(x))}
map{λx. hyponyms[spa](antonyms[spa](x))}	map{λx. place of death(named after(x))}	map{λx. union(mother(x), named after(x))}
map{λx. hyponyms[spa](entailments[spa](x))}	map{λx. place of death(x)}	map{λx. union(occupation(x), place of death(x))}
map{λx. hyponyms[spa](x)}	map{λx. position held(influenced by(x))}	map{λx. union(place of birth(x), position held(x))}
map{λx. influenced by(child(x))}	map{λx. position held(mother(x))}	map{λx. union(place of birth(x), position played on team(x))}
map{λx. influenced by(creator(x))}	map{λx. position held(x)}	map{λx. union(place of birth(x), record label(x))}
map{λx. influenced by(developer(x))}	map{λx. position played on team(father(x))}	map{λx. union(place of death(x), position held(x))}
map{λx. influenced by(father(x))}	map{λx. position played on team(named after(x))}	map{λx. union(place of death(x), position played on team(x))}
map{λx. influenced by(head of state(x))}	map{λx. position played on team(x)}	map{λx. union(place of death(x), record label(x))}
map{λx. influenced by(influenced by(x))}	map{λx. random-seed0[eng](x)}	map{λx. union(random-seed0[eng](x), antonyms[eng](x))}
map{λx. influenced by(owned by(x))}	map{λx. random-seed1[eng](x)}	map{λx. union(random-seed0[eng](x), entailments[eng](x))}
map{λx. influenced by(performer(x))}	map{λx. random-seed2[eng](x)}	map{λx. union(random-seed0[eng](x), hyponyms[eng](x))}
map{λx. influenced by(x)}	map{λx. random-seed3[eng](x)}	map{λx. union(random-seed0[eng](x), synonyms[eng](x))}
map{λx. intersection(antonyms[eng](x), entailments[eng](x))}	map{λx. record label(child(x))}	map{λx. union(random-seed1[eng](x), antonyms[eng](x))}
map{λx. intersection(antonyms[eng](x), hyponyms[eng](x))}	map{λx. record label(father(x))}	map{λx. union(random-seed1[eng](x), entailments[eng](x))}
map{λx. intersection(antonyms[eng](x), synonyms[eng](x))}	map{λx. record label(influenced by(x))}	map{λx. union(random-seed1[eng](x), hyponyms[eng](x))}
map{λx. intersection(antonyms[spa](x), entailments[spa](x))}	map{λx. record label(mother(x))}	map{λx. union(random-seed1[eng](x), synonyms[eng](x))}
map{λx. intersection(antonyms[spa](x), hyponyms[spa](x))}	map{λx. record label(x)}	map{λx. union(random-seed2[eng](x), antonyms[eng](x))}
map{λx. intersection(antonyms[spa](x), synonyms[spa](x))}	map{λx. sex or gender(x)}	map{λx. union(random-seed2[eng](x), entailments[eng](x))}
map{λx. intersection(entailments[eng](x), hyponyms[eng](x))}	map{λx. subclass of(x)}	map{λx. union(random-seed2[eng](x), hyponyms[eng](x))}
map{λx. intersection(entailments[eng](x), synonyms[eng](x))}	map{λx. synonyms[eng](x)}	map{λx. union(random-seed2[eng](x), synonyms[eng](x))}
map{λx. intersection(entailments[spa](x), hyponyms[spa](x))}	map{λx. synonyms[spa](x)}	map{λx. union(random-seed3[eng](x), antonyms[eng](x))}
map{λx. intersection(entailments[spa](x), synonyms[spa](x))}	map{λx. translate[eng->spa](antonyms[eng](x))}	map{λx. union(random-seed3[eng](x), entailments[eng](x))}
map{λx. intersection(hyponyms[eng](x), synonyms[eng](x))}	map{λx. translate[eng->spa](entailments[eng](x))}	map{λx. union(random-seed3[eng](x), hyponyms[eng](x))}
map{λx. intersection(hyponyms[spa](x), synonyms[spa](x))}	map{λx. translate[eng->spa](hyponyms[eng](x))}	map{λx. union(random-seed3[eng](x), synonyms[eng](x))}

Table 7: Full list of sequential compositional tasks in TASKBENCH500, organized by composition type.

filter{λx. is-POS-adjective[eng](x)}	filter{λx. is-POS-adverb[eng](x)}	filter{λx. is-POS-noun[eng](x)}	filter{λx. is-POS-verb[eng](x)}	filter{λx. is-sentiment-negative[eng](x)}
filter{λx. is-POS-adverb[eng](x)}	filter{λx. is-POS-noun[eng](x)}			

map{λx. antonyms[eng](x)}(filter{λx. is-POS-adjective[eng](x)})	map{λx. random-seed0[eng](x)}(filter{λx. is-POS-adjective[eng](x)})	map{λx. random-seed3[eng](x)}(filter{λx. is-POS-adjective[eng](x)})
map{λx. antonyms[eng](x)}(filter{λx. is-POS-adverb[eng](x)})	map{λx. random-seed0[eng](x)}(filter{λx. is-POS-adverb[eng](x)})	map{λx. random-seed3[eng](x)}(filter{λx. is-POS-adverb[eng](x)})
map{λx. antonyms[eng](x)}(filter{λx. is-POS-noun[eng](x)})	map{λx. random-seed0[eng](x)}(filter{λx. is-POS-noun[eng](x)})	map{λx. random-seed3[eng](x)}(filter{λx. is-POS-noun[eng](x)})
map{λx. antonyms[eng](x)}(filter{λx. is-POS-verb[eng](x)})	map{λx. random-seed0[eng](x)}(filter{λx. is-POS-verb[eng](x)})	map{λx. random-seed3[eng](x)}(filter{λx. is-POS-verb[eng](x)})
map{λx. antonyms[eng](x)}(filter{λx. is-sentiment-negative[eng](x)})	map{λx. random-seed0[eng](x)}(filter{λx. is-sentiment-negative[eng](x)})	map{λx. random-seed3[eng](x)}(filter{λx. is-sentiment-negative[eng](x)})
map{λx. antonyms[eng](x)}(filter{λx. is-sentiment-neutral[eng](x)})	map{λx. random-seed0[eng](x)}(filter{λx. is-sentiment-neutral[eng](x)})	map{λx. random-seed3[eng](x)}(filter{λx. is-sentiment-neutral[eng](x)})
map{λx. antonyms[eng](x)}(filter{λx. is-sentiment-positive[eng](x)})	map{λx. random-seed0[eng](x)}(filter{λx. is-sentiment-positive[eng](x)})	map{λx. random-seed3[eng](x)}(filter{λx. is-sentiment-positive[eng](x)})
map{λx. entailments[eng](x)}(filter{λx. is-POS-adjective[eng](x)})	map{λx. random-seed1[eng](x)}(filter{λx. is-POS-adjective[eng](x)})	map{λx. synonyms[eng](x)}(filter{λx. is-POS-adjective[eng](x)})
map{λx. entailments[eng](x)}(filter{λx. is-POS-adverb[eng](x)})	map{λx. random-seed1[eng](x)}(filter{λx. is-POS-adverb[eng](x)})	map{λx. synonyms[eng](x)}(filter{λx. is-POS-adverb[eng](x)})
map{λx. entailments[eng](x)}(filter{λx. is-POS-noun[eng](x)})	map{λx. random-seed1[eng](x)}(filter{λx. is-POS-noun[eng](x)})	map{λx. synonyms[eng](x)}(filter{λx. is-POS-noun[eng](x)})
map{λx. entailments[eng](x)}(filter{λx. is-POS-verb[eng](x)})	map{λx. random-seed1[eng](x)}(filter{λx. is-POS-verb[eng](x)})	map{λx. synonyms[eng](x)}(filter{λx. is-POS-verb[eng](x)})
map{λx. entailments[eng](x)}(filter{λx. is-sentiment-negative[eng](x)})	map{λx. random-seed1[eng](x)}(filter{λx. is-sentiment-negative[eng](x)})	map{λx. synonyms[eng](x)}(filter{λx. is-sentiment-negative[eng](x)})
map{λx. entailments[eng](x)}(filter{λx. is-sentiment-neutral[eng](x)})	map{λx. random-seed1[eng](x)}(filter{λx. is-sentiment-neutral[eng](x)})	map{λx. synonyms[eng](x)}(filter{λx. is-sentiment-neutral[eng](x)})
map{λx. entailments[eng](x)}(filter{λx. is-sentiment-positive[eng](x)})	map{λx. random-seed1[eng](x)}(filter{λx. is-sentiment-positive[eng](x)})	map{λx. synonyms[eng](x)}(filter{λx. is-sentiment-positive[eng](x)})
map{λx. hyponyms[eng](x)}(filter{λx. is-POS-adjective[eng](x)})	map{λx. random-seed2[eng](x)}(filter{λx. is-POS-adjective[eng](x)})	map{λx. translate[eng->spa](x)}(filter{λx. is-POS-adjective[eng](x)})
map{λx. hyponyms[eng](x)}(filter{λx. is-POS-adverb[eng](x)})	map{λx. random-seed2[eng](x)}(filter{λx. is-POS-adverb[eng](x)})	map{λx. translate[eng->spa](x)}(filter{λx. is-POS-adverb[eng](x)})
map{λx. hyponyms[eng](x)}(filter{λx. is-POS-noun[eng](x)})	map{λx. random-seed2[eng](x)}(filter{λx. is-POS-noun[eng](x)})	map{λx. translate[eng->spa](x)}(filter{λx. is-POS-noun[eng](x)})
map{λx. hyponyms[eng](x)}(filter{λx. is-POS-verb[eng](x)})	map{λx. random-seed2[eng](x)}(filter{λx. is-POS-verb[eng](x)})	map{λx. translate[eng->spa](x)}(filter{λx. is-POS-verb[eng](x)})
map{λx. hyponyms[eng](x)}(filter{λx. is-sentiment-negative[eng](x)})	map{λx. random-seed2[eng](x)}(filter{λx. is-sentiment-negative[eng](x)})	map{λx. translate[eng->spa](x)}(filter{λx. is-sentiment-negative[eng](x)})
map{λx. hyponyms[eng](x)}(filter{λx. is-sentiment-neutral[eng](x)})	map{λx. random-seed2[eng](x)}(filter{λx. is-sentiment-neutral[eng](x)})	map{λx. translate[eng->spa](x)}(filter{λx. is-sentiment-neutral[eng](x)})
map{λx. hyponyms[eng](x)}(filter{λx. is-sentiment-positive[eng](x)})	map{λx. random-seed2[eng](x)}(filter{λx. is-sentiment-positive[eng](x)})	map{λx. translate[eng->spa](x)}(filter{λx. is-sentiment-positive[eng](x)})

Table 8: Full list of sequential compositional tasks in TASKBENCH500, organized by composition type. *(Continued from previous page.)*