Learning from Task Descriptions

Orion Weller*[1], Nicholas Lourie[2], Matt Gardner[2], Matthew E. Peters[2]
[1]Brigham Young University
[2]Allen Institute for Artificial Intelligence
orionw@byu.edu,{nicholasl,mattg,matthewp}@allenai.org

Abstract

Typically, machine learning systems solve new tasks by training on thousands of examples. In contrast, humans can solve new tasks by reading some instructions, with perhaps an example or two. To take a step toward closing this gap, we introduce a framework for developing NLP systems that solve new tasks after reading their descriptions, synthesizing prior work in this area. We instantiate this framework with a new English language dataset, ZEST, structured for task-oriented evaluation on unseen tasks. Formulating task descriptions as questions, we ensure each is general enough to apply to many possible inputs, thus comprehensively evaluating a model's ability to solve each task. Moreover, the dataset's structure tests specific types of systematic generalization. We find that the state-of-the-art T5 model achieves a score of 12% on ZEST, leaving a significant challenge for NLP researchers.[1]

1 Introduction

The dominant paradigm in supervised NLP today is learning from examples, where machine learning algorithms are trained using a large set of task-specific input-output pairs. In contrast, humans learn to perform the same task by reading a description, after which they are able to perform the task in a zero-shot manner—indeed, this is how crowd-sourced NLP datasets are constructed. In this paper, we argue that learning from task descriptions in this way is a necessary attribute of a general purpose NLP system, and we propose it as a new paradigm to train and test NLP systems.

Recent work in NLP has shown significant progress in learning tasks from examples. Large pretrained language models have dramatically improved performance on standard benchmarks (Peters et al., 2018; Devlin et al., 2019; Raffel et al., 2019) and have shown promising results in zero shot prediction by leveraging their language understanding capabilities (Levy et al., 2017; Zhou et al., 2018; Yin et al., 2019).

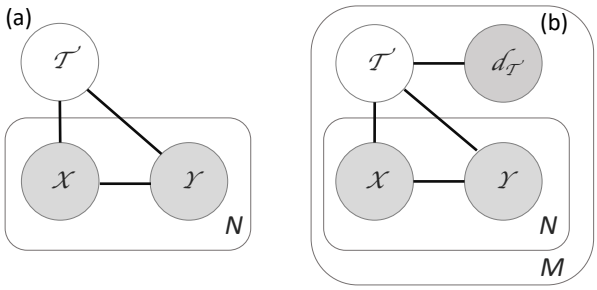

Figure 1: Comparison of (a) supervised learning from examples with observed input X, output Y, corresponding to an unobserved task τ (b) our proposed method of learning from task descriptions where systems can make inferences about unseen tasks τ given a natural language description d_τ.

Despite this progress, there are many serious issues that come with learning from examples. There is an almost infinite number of tasks that a person might wish to solve with a general-purpose NLP system. Learning to solve these tasks by reading a description instead of observing a collection of examples would solve the problem of having to create training sets for each language task. Such a system would also be more accessible to practitioners and domain experts in other fields, who could describe their tasks and solve them, opening up new avenues of research where it is expensive or infeasible to gather training data.

Additionally, we find that current supervised learning techniques partly achieve their success due to memorizing uninteresting aspects of the training distribution (Gururangan et al., 2018; Geva et al., 2019; Gardner et al., 2020). Teaching a system to learn a task from the description alone would alleviate these biases, as new training data would not be needed to learn a novel task.

*Work done while at the Allen Institute for AI.
[1]Data, evaluation code, baseline models, and leaderboard at https://allenai.org/data/zest

Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, pages 1361–1375,
November 16–20, 2020.

In this paper, we synthesize prior approaches to zero-shot learning in NLP and provide a formal framework for thinking about the zero-shot prediction problem. We show that previous zero-shot approaches are limited in both scope of application and rigour of evaluation. For example, while prior work has used zero-shot prediction for text classification, entity typing, and relation extraction, we push this to the more complex task of slot filling.

We instantiate our formalism in an English language dataset, ZEST (**ZE**ro **S**hot learning from **T**ask descriptions), that is formatted similarly to reading comprehension datasets, in that we formulate task descriptions as questions and pair them with paragraphs of text. We choose this format as it provides a natural way to crowdsource data. This zero-shot dataset differs from typical reading comprehension datasets, however, in that each task description is paired with twenty different passages, and we evaluate a model's ability to solve the task, not just give the correct answer for a single (question, passage) pair. That is, given a question, a model produces some decision function f, and it is this function which we comprehensively evaluate on many different inputs. We also carefully select axes on which to evaluate the generalization of a model to different kinds of task descriptions, changing task descriptions in specific ways to systematically push the field towards more interesting and complex task descriptions.

We evaluate models based on recent state-of-the-art sequence to sequence architectures, which seem most suited to the task of zero shot prediction in this setting. We find that our best model based on T5 (Raffel et al., 2019) achieves a score of only 12% on this data, leaving a significant gap to our human performance estimate of 42%. Zero shot learning from complex task descriptions remains a significant challenge for current NLP systems.

2 Learning from task descriptions

This section describes our framework for enabling zero-shot generalization to unseen tasks, and relates it to prior work.

2.1 Learning from examples

Consider the supervised learning setting[2] where the goal is to learn a function $y = f_\theta(x)$, with trainable parameters θ, for a particular task. We define the task τ as:

- a definition for the sets of allowable inputs $x \in \mathcal{X}$, outputs $y \in \mathcal{Y}$, and,
- a probability distribution $p_\tau(x, y)$.

In text classification, for example, $\mathcal{X}$ is natural language text and $\mathcal{Y}$ is a categorical label from one of C classes. In the single task setting, the function f is learned by collecting a dataset of labeled examples $\mathcal{D} = \{(x_1, y_1), \ldots (x_N, y_N)\}$ sampled from $p_\tau(x, y)$ (see Fig. 1a). We call this "learning from examples". Crucially, once $\mathcal{D}$ is constructed, the underlying task definition is discarded, assumed to be captured in the labeled (x_i, y_i) pairs.

There are many ways to sample from $p_\tau(x, y)$ to create a dataset. One approach, in cases such as language modeling where p_τ is defined by a set of rules, just applies the rules to raw text. Another popular approach uses human annotation. In this case, the most common strategy factorizes $p_\tau(x, y) = p_\tau(y|x)p_\tau(x)$, samples from $p_\tau(x)$ via some method (e.g. collecting text from the domain of interest), and uses a natural language task description, d_τ, to describe $p_\tau(y|x)$. The description is shown to human annotators who use it to compute $\arg\max_{y \in \mathcal{Y}} p(y|x_0)$ for a given x_0.

2.2 Learning from task descriptions

The largest downside to learning from examples is that every new task requires collecting a new dataset to learn a new function $f_\theta(x)$ for the task. This approach also discards the task definition after the labeled dataset is constructed, despite the fact that the task definition carries all of the information necessary for a human to solve the task. Moreover, it holds the task constant at test time (except in certain limited cases, see Sec. 2.4).

Our proposed framework, which we call "learning from task descriptions", removes these restrictions. First, instead of discarding the task definition, we provide a natural language description of it to the model, in addition to the input x. Second, by providing the model with the task description, we expect it to generalize to *unseen tasks* at test time in a zero-shot way.

These modifications shift the learning problem from fitting a probability distribution in the learning from examples approach, to understanding the semantics of a task description in order to apply it

[2]This setting also includes popular self-supervised objectives such as autoregressive or masked language modeling.

to a given input in the learning from task descriptions approach. Successfully building a model to perform in this manner would open up a wide range of NLP applications whereby one could simply construct an NLP system by describing the desired output in natural language.

Our proposed framework is illustrated in Fig. 1b. In contrast to learning from examples, we assume the task description d_τ is observed for M different tasks, and that each of these tasks has some number N of observed (x_i, y_i) pairs.

2.3 Task competence

In order to test whether a system can adequately perform an unseen task, we propose a new evaluation metric as follows. Traditional evaluation metrics in supervised learning are averages over instance-level metrics, that is, they perform independent computation on individual (x, y) pairs and aggregate them across a dataset to produce a summary score. As we are interested in assessing whether a model can competently perform a task from its description, we instead first evaluate whether a model can perform each individual task using the entire *set* of (x, y) pairs for a given task, and then report averages over all tasks.

Formally, a dataset with M tasks can be viewed as the concatenation of M different N_j sized datasets, $\mathcal{D}_j = \{(x_1, y_1), \ldots (x_{N_j}, y_{N_j})\}$, one for each task. We assume each task has an associated metric $\mu_j(\mathcal{D}_j, f_\theta) \in \mathbb{R}$, which is used to compute the model performance for task τ_j on $\mathcal{D}_j$ for the model represented by f_θ. For simplicity, we assume each metric is such that larger values indicate better performance[3]. Then, for a given level of competence c_j for task τ_j, we say that the model can perform the task if $\mu_j \geq c_j$. The final model competence metric is the average individual task competence over the dataset, $c = \frac{1}{M} \sum_j \mathbb{1}(\mu_j \geq c_j)$, where $\mathbb{1}$ is the indicator function. In the special case where c_j has the same threshold T for all j, we write "C@T" to represent the competence at T.

As a concrete example of this metric, consider the simple case where all M tasks are binary classification (so that unseen classes correspond to unseen tasks). If we adopt accuracy as the metric for all tasks, and set c_j to 90% for all j then a C@90 of 72% indicates that the model is able to successfully classify unseen inputs x into a set of unseen classes $\mathcal{Y}$ with at least 90% accuracy, for 72% of the unseen tasks τ.

[3]This can be achieved by rescaling if necessary.

2.4 Discussion

Prior researchers have recognized the limitations of learning from examples, and have worked to address some of them. Our proposed framework builds upon and generalizes much of this work.

Zero-shot learning (Chang et al., 2008; Socher et al., 2013; Norouzi et al., 2013) asks systems to generalize to unseen classes at test time. In this approach, the task is the same at both train and test time—models are only asked to generalize to new classes. In terms of the graphical model in Fig. 1, prior work attaches a natural language description to some new y_i at test time. In contrast, our approach asks models to generalize to entire unseen *tasks*, attaching the natural language description to the task variable τ. Zero-shot learning has been widely adopted including for classification (Dauphin et al., 2013), entity typing (Ma et al., 2016; Zhou et al., 2018) and relation extraction (Levy et al., 2017; Shi and Lin, 2019).

More closely related to our approach are the zero-shot experiments in Radford et al. (2019); Brown et al. (2020) that provide a generative language model with a prompt (that could be viewed as a type of task description) and asks for a completion. This is similar to the observation in Petroni et al. (2019) that it is possible to extract knowledge graph relationships from large language models with an appropriate prompt. ZEST provides a benchmark dataset for systematically measuring how well models can generalize to many tasks in the zero-shot setting.

Multitask learning (Caruana, 1997; Collobert and Weston, 2008) seeks to learn a single model that can solve multiple tasks simultaneously, similar to our framework that seeks to learn a model that can solve many tasks. However, in multitask learning each task is learned from examples, and the model is not able to generalize to unseen tasks. This is also the case for newer control code type approaches (Raffel et al., 2019; Keskar et al., 2019) to multitask learning, where the task is encoded as short string, often containing no information other than a largely meaningless identifier.

There are also connections between our proposed framework and tasks such as natural language inference (NLI) or reading comprehension (RC), where two natural language inputs (a

premise and a hypothesis for NLI, and a question and passage for RC) are used to predict some output. In our case, we have two observed variables, x and d_τ, which influence the prediction of the output y (Fig. 1). Indeed, the baseline model that we discuss in Section 5 takes a similar approach to NLI and RC and jointly models the two textual inputs. This correspondence has been used in prior work, where Yin et al. (2019) used a model pretrained on MNLI (Williams et al., 2018) to perform zero-shot text classification. A key difference, however, is that hypotheses in NLI and questions in RC are typically only paired with single inputs. In fact, they typically only make sense for a single input, and thus it is hard to characterize these narrow questions as "task descriptions".

Lastly, the problem of learning from task descriptions is fundamentally one of translating a natural language description into some executable function that can operate on arbitrary inputs. This problem has been well-studied for narrow domains in the semantic parsing literature (Zelle and Mooney, 1996; Zettlemoyer and Collins, 2005; Liang et al., 2011; Andreas et al., 2013), though the input is typically a single static database, not arbitrary natural language text. Attempts to generalize semantic parsing to more open domains are still nascent (Chen et al., 2020; Gupta et al., 2020).

3 Instantiating the Framework

Section 2 showed a framework for training and testing a general purpose system that could perform unseen NLP tasks. An ideal system in this framework would be able to read the descriptions of the tasks in the GLUE suite (Wang et al., 2019) and perform well with no additional training. However, this goal is far beyond the current capabilities of today's models. In order to make progress, we must break down the problem into manageable steps. In this section we outline the scope that we envision for a reasonable NLP-focused dataset that can push forward the current state of learning from task descriptions, without being so challenging as to be out of reach. Sec. 4 describes the data collection process for ZEST, our new English benchmark built following this scope.

To define the scope, we begin by considering the types of applications a model that could successfully learn from task descriptions might enable. The largest bottleneck in building NLP applications today is collecting labeled data. Our framework would eliminate this step, making it possible to build ad hoc NLP applications to easily filter, categorize, or extract structured information from corpora. For example, when planning a camping trip, one might want to know "What are the names of all the campgrounds and their locations?" that are listed in a collection of documents, which specifies an ad hoc request to return all examples of the `located_at` relationship between the `campground` and `location` entity types. Accordingly, it's important to include examples of the basic task building blocks of such a system: classification, typed entity extraction, and relation extraction in a benchmark dataset. In doing so, it would unify the prior work in zero-shot NLP (Sec. 2.4) that has focused on just a single task, and require a single model to be able to handle any of these tasks at test time, instead of separate models for each task.

More concretely, as each task τ defines a set of allowable outputs $y \in \mathcal{Y}$, we can mix multiple output sets $\mathcal{Y}$ in a single dataset as long as the output set is specified in the task description. ZEST includes the most common output sets: discrete classes, lists of (optionally) typed spans from the input, and relationships between spans. Examples of each are shown in Table 1, where it is clear from the task description which output $\mathcal{Y}$ is expected. In addition, we also include the `NA` output (Rajpurkar et al., 2018), signifying that it is not possible to solve the task given the input x. For example, if the task asks a model to extract campground names but the input is an unrelated news article, the output is `NA`. Being able to correctly identify unsolvable tasks is important in a practical setting where it is not reasonable to expect every possible task to be solvable with every possible input.

To move beyond aggregating existing approaches into a single dataset, recall that in our framework observing the task description d_τ in addition to the input x allows us to test a model's generalization relative to four variables: x, y, τ, and d_τ (Fig. 1). Motivated by this observation, we propose an approach that systematically varies the task descriptions and inputs while controlling for other sources of variability in order to test whether a system can generalize in multiple ways. To implement this idea, we begin by collecting a set of task descriptions, d_τ, inputs x, and associated outputs, y. This *base* group of instances already allows us to test performance of unseen tasks on un-

Generalization	Question	Input Passage (shortened)	Answer
Base	Can I hike to a waterfall at this national park?	... Yet here at Whiskeytown NRA, we encourage you to chase waterfalls - go visit them! Whiskeytown has four major waterfalls ...	Yes
Paraphrase	Is there a waterfall to hike to at this national park?	(same as above)	Yes
Semantic Flips	Can I hike to a canyon at this national park?	... descending 1,300 feet (396 m) past a large alcove, the trail meanders in a wide canyon ...	Yes
Composition	What time of year is best to see the popular waterfalls in this national park?	... Two viewing platforms provide the best view of Great Falls. This overlook is the last place that the Falls can be viewed ...	NA
Output Structure	What waterfall hikes are there in this national park and are they wheelchair accessible?	... Bridalveil Fall is often the first waterfall you'll see when entering ... Although paved, this is trail is not wheelchair accessible due to its grade.	[{"waterfall hike":"Bridalveil Fall", "wheelchair accessible": "No"}]

Table 1: Example instances from ZEST. The composition question is combined with "What are the popular tourist spots in this national park?" We chose to format the relation extraction questions as JSON, see Section 5.2 for details.

seen input. We further augment it with four types of controlled generalization: paraphrase, semantic flips, composition, and output structure. Examples of each type of generalization are given in Table 1.

Paraphrase We can test generalization to changes in the task description d_τ while keeping the task τ fixed by paraphrasing the description. By also fixing x, we can use these paraphrases to test whether a model consistently predicts the correct output given the same input and underlying task. As we collect applicable inputs x for a task using a retrieval mechanism given the task description (Section 4), this also adds some lexical distance between the input and the description, to avoid simple lexical shortcuts to solving the task (Gardner et al., 2019).

Semantic flips Closely contrasting examples have long provided an effective means of evaluation in NLP (Levesque et al., 2012; Sennrich, 2017), forcing a model to understand how small changes in inputs correspond to large changes in expected outputs. We take inspiration from this idea to include task description semantic flips, where a given task is modified in a minimal way (e.g. by changing a single word) to semantically change the meaning of the task. As the description is largely unchanged (including the output set $\mathcal{Y}$), this tests whether systems can distinguish between descriptions that are minimally changed.

Composition To further test whether systems can understand a task description, we can compose base tasks into new tasks with operators such as "and" and "or". By examining the performance difference between the base group of tasks and the compositionally generated group of tasks we can estimate the extent to which a system can compose tasks in a novel way.

Output structure We can also test whether models can generalize to unseen structured outputs $y_1 \in \mathcal{Y}$ where y_1 is not seen in the training set. Among the many ways to accomplish this, we chose a method that asks models to produce output equivalent to slot filling or n-ary relationship extraction in the zero-shot setting. In this case,

task descriptions correspond to a specification of an output structure that includes typed entity and relationship extraction where the entity types and relationships have not been seen in training.

4 Collecting ZEST

To illustrate our novel way of evaluating and framing the "learning from task descriptions" problem, we provide an empirical demonstration of where current systems fail by collecting a challenge dataset. We hope this will serve as a starting point for making progress towards this goal of learning from descriptions. In this section we describe our annotation efforts, which consist of our design for the dataset, as well as three crowdsourcing steps: collecting tasks (in question form), gathering relevant documents, and annotating answers for the (task, document) pairs.

4.1 Dataset Design

Our dataset consists of base task descriptions which are varied along the four areas of generalization found in Section 3, allowing us to systematically control for generalization across the different base tasks. We collect annotations for approximately 20 different input documents for each task so that we can calculate the competency metric.

The framework described in Section 2.4 applies to any task description, thus, it is agnostic to the specific format. In deciding how to format the task descriptions in ZEST we chose to use a question format for the tasks, as crowdsourcing annotations for questions is well established, and a QA format may potentially allow transfer from existing question answering datasets. We note however, that a declarative task description such as "return a list of hikes in the national park described in the document" fundamentally asks for the same information as the question "what are the hikes in this national park?" As a result, we will use the terms *task description* and *question* interchangeably when discussing our creation of ZEST.

4.2 Task Generation

As each question should apply to numerous documents, we used Mechanical Turk[4] to crowdsource common questions that someone might ask about three different domains: U.S. presidents, dog breeds, and U.S. national parks. We use multiple domains to include diversity in our tasks, choosing domains that have a multitude of entities to which a single question could be applied. Workers were asked to generate questions that could apply to any entity in that domain and we manually removed questions that contained duplicate meanings to maintain a rich semantic space. This left us with approximately 100 base task descriptions for each domain. These tasks were generated before gathering input documents, alleviating biases from having workers who had already seen the input passages.

We split these tasks into 50% test, 40% train, and 10% development. We then employed other workers to alter them along one of the four areas of generalization. For the paraphrase generation, we asked workers to paraphrase the text so that it retained its original meaning but had a different wording. For the semantic flip questions we asked the workers to keep as much of the task description the same as possible, but to make a slight change that would alter the meaning of the task. Composition tasks were created by randomly sampling three tasks from within each dataset split to combine, letting the worker choose two out of the three. Tasks for the output structure were created by expanding the base tasks to include multiple structured sub-tasks, using a custom built UI that automatically compiled workers' responses into JSON format.

Each task description created for a particular area of generalization followed its base task to the corresponding dataset split. Hence the test set contains its own unique base questions as well the derived questions for each area of generalization.

Statistic	Train	Dev	Test
(task, passage) pairs	10,766	2,280	11,980
Avg. passage words	121	122	122
Number of tasks	538	114	599
Avg. task len [words]	12.3	12.2	11.8
NA percent	0.62	0.67	0.62
Classification Percent	0.46	0.49	0.44

Table 2: Summary Statistics for ZEST. Note that NA is the most frequent answer.

[4]We initially opened our crowdsourcing pipeline to the U.S. population on Mechanical Turk that had above a 99% acceptance rate with over 5000 completed HITs, but reduced this pool to only include workers who performed well on initial HITs.

4.3 Passage Retrieval

In order to gather a unique set of passages that pertain to a given question, we used Bing and Google Custom Search engines, focusing the results on a narrow subset of webpages. For U.S. Presidents, our queries were limited to results from Wikipedia pages (for all 45 presidents) as well as information contained on Whitehouse.gov, containing biographies and accomplishments for each President and First Lady. Similarly, we limited our queries of dog breeds to all 524 pages of Dog Breeds on Wikipedia. The U.S. National Park passages were retrieved from sub-pages of the National Parks website. On each of these domains, we ensured that no single entity garnered more than 5% of the total input documents. Details on how we used these search engines to gather the passages can be found in Appendix A and in our code.

4.4 Document Annotations

We paired the gathered task descriptions with their respective passages and employed our expert workers from Mechanical Turk to annotate the answers. We had three workers annotate each (task, document) pair. For the tasks that could be answered with a yes or no response, final answers were chosen by taking the majority answer. For tasks that involved extracting information from the passage, we used the answer that was the subset of the other answers, preferring shorter responses over longer responses. 25,026 (task, input, answer) triples, with a total of 1251 task descriptions split across the three domains. These tasks were distributed as 45% extraction, 45% classification and 10% mixed (due to the output structure tasks). More summary statistics can be found in Table 2. Our annotation costs were approximately 9,000 USD.

5 Establishing a Baseline

This section describes our baseline model results.

5.1 Evaluation

Due to class imbalance, we adopt F1 as the metric when computing the task competency (Sec. 2.3). However, to account for partial overlap between model and gold answers, we modify the precision P and recall R as follows. Each task τ has a number of instances (x_i, y_i). For each instance, we compute a partial overlap score s_i that includes an output-type aware[5] best alignment between the model and gold answers and scores individual elements with a word overlap based method. This is similar to common practice in QA evaluation, extended to handle ZEST's output types. Then, with NA as the negative class, we compute $P = \sum_i s_i/m^+$, $R = \sum_i s_i/g^+$ where m^+ and g^+ are the total model predicted positive (not-NA) and gold positive instances.

We take each task's F1 score and evaluate the competency metric for each task, reporting these scores in our final results. Additionally, when tasks are closely related we use a more stringent *consistency* metric (Gardner et al., 2020) that computes whether a model is competent in both tasks at the same time. For paraphrases and semantic flips, our C@T metrics only count a model as competent for a task if it is competent for both the base task description and the changed task description. This helps to avoid giving the model credit for artificially simple decision boundaries that only accidentally solve the task.

5.2 Modeling

For baselines, we adopt two recent state-of-the-art models, T5 (Raffel et al., 2019) and BART (Lewis et al., 2020), both because of their positions on top of popular NLP leaderboards and their text-to-text nature. Beyond training on ZEST alone, we also trained T5 using multitask learning (MTL) with a combination of other QA datasets to test transfer to ZEST: BoolQ (Clark et al., 2019), MultiRC (Khashabi et al., 2018), ReCoRD (Zhang et al., 2018), and SQuAD (Rajpurkar et al., 2016).

Data Preprocessing To prepare each task's instances for the model, we prepended "zeroshot question: " to the task description and "zeroshot context: " to the document, then joined these two parts together with whitespace. For output structure generalization, we formatted the answers as JSON to enable more complex zero-shot relation extraction tasks. Thus, the models output answers as both text and JSON, in a seq-to-seq fashion, depending on the question type. When the question calls for JSON, we deserialize and evaluate it, counting deserialization failures as errors. See Appendix B for more on data preprocessing.

Training & Hyper-parameters For T5 11B, our best baseline, training used input and output

[5] ZEST includes strings, sets of strings, lists of dicts, and three discrete classes (Yes/No/NA) as valid output types.

	Dev			Test		
	Mean	C@75	C@90	**Mean**	C@75	C@90
BART-large ZEST only	40	13	8	38	11	4
T5-11B ZEST only	56	32	12	55	28	11
T5-11B ZEST w/MTL	56	35	14	56	28	12
Human Estimate				74	61	42

Table 3: Overall performance of baseline models showing the mean F1 and competency at 75% and 90%. Our best model, a T5 model with multi-task learning from other QA datasets (Section 5.2), is only able to perform 12% of unseen tasks at 90% F1, compared to a human estimate of 42% of tasks at 90% competency.

	Dev			Test		
Generalization Type	**Mean**	C@75	C@90	**Mean**	C@75	C@90
Base	71	48	16	63	43	22
Paraphrase	64	36	12	56	32	16
Composition	66	44	22	65	41	15
Semantic Flips	54	27	9	47	18	5
Output Structure	33	20	10	47	10	3
Overall w/MTL	56	35	14	56	28	12

Table 4: Detailed T5-11B results for ZEST with multi-task learning using other QA datasets (Section 5.2).

Input	**Mean**	C@75	C@90
Full data	56	32	12
Question only	12	10	7
Context only	1	1	1

Table 5: T5-11B ablation results on the development set using the full dataset, question only and context only. Only the overall results are shown. The context only model predicted NA for each instance.

sequence lengths of 512, a batch size of 32, and grid searched four different learning rates (5e-4, 1e-3, 2e-3, and 4e-3). See Appendix C for BART and other T5 details.

5.3 Results

We present our overall results on ZEST, an ablation using T5 to probe for annotation artifacts (Gururangan et al., 2018), and an error analysis breaking down common mistakes.

Baseline Performance Table 3 shows the performance of the baselines on ZEST, as well as an estimate of human performance.[6] We report mean F1 across all *instances* in the data, ignoring their grouping into tasks, as well as our proposed C@T metric, for T $\in \{75, 90\}$. The best T5-11B model has mean performance of 56% on the development set, while the BART model has lower scores. Moreover, when we evaluate *task competence*, we see these models only rarely successfully solve the whole task well. For C@90, the T5 model's overall score is only 12% on the test set. Multitasking ZEST with other QA datasets only slightly improved results. Table 4 shows a detailed breakdown of performance across generalization type for the T5 model with multi-tasking. Detailed results for BART are in the Appendix. Model performance decreases as the generalization difficulty increases from the Base level to Output Structure. Consistently recovering models from task descriptions alone remains a significant challenge.

Annotation Artifacts & Ablations Table 5 shows ablations on the dev set using T5, illustrating that both the question and context are needed for the model to perform well, as one would expect. We see that in the context only ablation, the model predicted NA (majority class) for all instances, showing that there were not any systematic biases in the passages alone that the model could exploit. The context only F1 is non-zero due the fact that one task had all NA answers, which is

[6] Computed by having an author label answers to 55 tasks from the test set.

Error	Question	Input Passage (shortened)	Predicted	Correct
Recall (30%)	Did this president get a graduate degree?	... at Harvard University, where he earned an M.A. in economics ...	N/A	Yes
Precision (37%)	Are the volcanoes in this national park dormant?	... Dormant: A volcano that is inactive or resting, but is likely to erupt again in the near future. Extinct: A volcano that has stopped erupting ...	Yes	NA
Partial (9%)	What kind of trout can be found at this national park?	... The presence of non-native brown trout has the potential to impact brook trout and other native fish populations within several of the park's premier large streams ...	Brown trout	Brown trout,brook trout
Other (24%)	Was this dog breed accepted in the american kennel club in the last twenty years?	... The Cavalier would go on to be recognized by the American Kennel Club in 1995 ...	No	Yes

Table 6: Error distribution of the baseline model. Recall errors are when the model incorrectly predicts N/A; precision errors are when the model should have predicted N/A, but didn't; partial answers are when the model failed to predict all of the members of a list. Other common errors included failing to apply reasoning to answer a question, and predicting the wrong key names when producing JSON outputs.

counted as competent by convention.

Error Analysis In order to more clearly understand where these models fail, we examined 100 instances of model errors and categorized them. The most frequent errors were when the model failed to recognize the answer (30% of the time) or predicted something when the answer was NA (37%). We provide detailed examples and descriptions in Table 6. Interestingly, the model failed to output parseable JSON on only 1.5% of all structure questions in the test set and generated a JSON structure format for only 0.008% of non-structure questions, showing strong results for learning the format for outputting the complex relationships.

6 Conclusion

We introduced a framework for creating general purpose NLP systems that can solve tasks from natural language descriptions, synthesizing and extending previous work in zero-shot learning. To make progress toward this goal, we create a dataset, ZEST, that rigorously evaluates how well a model truly understands each task. The dataset is designed to test models' ability to systematically generalize across four different areas. State-of-the-art performance on ZEST is 12%, leaving much room for future improvement.

While we have been focused on zero shot learning from task descriptions, our framework also permits few-shot scenarios where a task description is given along with a handful of examples, making meta-learning approaches applicable. This is an interesting avenue for future work, for which ZEST should also be useful. To facilitate future work, we make our models, code, and data available at https://allenai.org/data/zest.

Acknowledgements

The authors acknowledge helpful feedback from anonymous reviewers and the AllenNLP team. TPU compute used in this work was provided by Google through TensorFlow Research Cloud (TFRC). This research was funded in part by the NSF under awards IIS-1817183 and CNS-1730158.

References

Jacob Andreas, Andreas Vlachos, and Stephen Clark. 2013. Semantic parsing as machine translation. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 47–52, Sofia, Bulgaria. Association for Computational Linguistics.

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

Rich Caruana. 1997. Multitask learning. *Machine learning*, 28(1):41–75.

Ming-Wei Chang, Lev-Arie Ratinov, Dan Roth, and Vivek Srikumar. 2008. Importance of semantic representation: Dataless classification. In *AAAI*.

Xinyun Chen, Chen Liang, Adams Wei Yu, Denny Zhou, Dawn Song, and Quoc V. Le. 2020. Neural symbolic reader: Scalable integration of distributed and symbolic representations for reading comprehension. In *International Conference on Learning Representations*.

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. *arXiv preprint arXiv:1905.10044*.

Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th international conference on Machine learning*, pages 160–167.

Pradeep Dasigi, Nelson F Liu, Ana Marasovic, Noah A Smith, and Matt Gardner. 2019. Quoref: A reading comprehension dataset with questions requiring coreferential reasoning. *arXiv preprint arXiv:1908.05803*.

Yann N Dauphin, Gokhan Tur, Dilek Hakkani-Tur, and Larry Heck. 2013. Zero-shot learning for semantic utterance classification. *arXiv preprint arXiv:1401.0509*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *NAACL-HLT*.

Dheeru Dua, Yizhong Wang, Pradeep Dasigi, Gabriel Stanovsky, Sameer Singh, and Matt Gardner. 2019. Drop: A reading comprehension benchmark requiring discrete reasoning over paragraphs. *arXiv preprint arXiv:1903.00161*.

Matt Gardner, Yoav Artzi, Victoria Basmova, Jonathan Berant, Ben Bogin, Sihao Chen, Pradeep Dasigi, Dheeru Dua, Yanai Elazar, Ananth Gottumukkala, Nitish Gupta, Hanna 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 Quan Zhang, and Ben Zhou. 2020. Evaluating nlp models via contrast sets. *ArXiv*, abs/2004.02709.

Matt Gardner, Jonathan Berant, Hannaneh Hajishirzi, Alon Talmor, and Sewon Min. 2019. On making reading comprehension more comprehensive. In *Proceedings of the 2nd Workshop on Machine Reading for Question Answering*, pages 105–112, Hong Kong, China. Association for Computational Linguistics.

Mor Geva, Yoav Goldberg, and Jonathan Berant. 2019. Are we modeling the task or the annotator? an investigation of annotator bias in natural language understanding datasets. 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 1161–1166, Hong Kong, China. Association for Computational Linguistics.

Nitish Gupta, Kevin Lin, Dan Roth, Sameer Singh, and Matt Gardner. 2020. Neural module networks for reasoning over text. In *International Conference on Learning Representations*.

Suchin Gururangan, Swabha Swayamdipta, Omer Levy, Roy Schwartz, Samuel Bowman, and Noah A. Smith. 2018. Annotation artifacts in natural language inference data. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 107–112, New Orleans, Louisiana. Association for Computational Linguistics.

Nitish Shirish Keskar, Bryan McCann, Lav Varshney, Caiming Xiong, and Richard Socher. 2019. CTRL - A Conditional Transformer Language Model for Controllable Generation. *arXiv preprint arXiv:1909.05858*.

Daniel Khashabi, Snigdha Chaturvedi, Michael Roth, Shyam Upadhyay, and Dan Roth. 2018. Looking beyond the surface:a challenge set for reading comprehension over multiple sentences. In *NAACL*.

Hector Levesque, Ernest Davis, and Leora Morgenstern. 2012. The winograd schema challenge. In *Thirteenth International Conference on the Principles of Knowledge Representation and Reasoning*.

Omer Levy, Minjoon Seo, Eunsol Choi, and Luke Zettlemoyer. 2017. Zero-shot relation extraction via reading comprehension. In *Proceedings of the 21st Conference on Computational Natural Language Learning (CoNLL 2017)*, pages 333–342, Vancouver, Canada. Association for Computational Linguistics.

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

Percy Liang, Michael Jordan, and Dan Klein. 2011. Learning dependency-based compositional semantics. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 590–599, Portland, Oregon, USA. Association for Computational Linguistics.

Yukun Ma, Erik Cambria, and Sa Gao. 2016. Label embedding for zero-shot fine-grained named entity typing. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 171–180.

Mohammad Norouzi, Tomas Mikolov, Samy Bengio, Yoram Singer, Jonathon Shlens, Andrea Frome, Greg S Corrado, and Jeffrey Dean. 2013. Zero-shot learning by convex combination of semantic embeddings. *arXiv preprint arXiv:1312.5650*.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *NAACL-HLT*.

Fabio Petroni, Tim Rocktäschel, Patrick Lewis, Anton Bakhtin, Yuxiang Wu, Alexander H. Miller, and Sebastian Riedel. 2019. Language models as knowledge bases? *ArXiv*, abs/1909.01066.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2019. Exploring the limits of transfer learning with a unified text-to-text transformer. *arXiv preprint arXiv:1910.10683*.

Pranav Rajpurkar, Robin Jia, and Percy Liang. 2018. Know what you don't know: Unanswerable questions for SQuAD. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 784–789, Melbourne, Australia. Association for Computational Linguistics.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100,000+ questions for machine comprehension of text. *arXiv preprint arXiv:1606.05250*.

Rico Sennrich. 2017. How grammatical is character-level neural machine translation? assessing MT quality with contrastive translation pairs. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 376–382, Valencia, Spain. Association for Computational Linguistics.

Noam Shazeer, Youlong Cheng, Niki Parmar, Dustin Tran, Ashish Vaswani, Penporn Koanantakool, Peter Hawkins, HyoukJoong Lee, Mingsheng Hong, Cliff Young, Ryan Sepassi, and Blake Hechtman. 2018. Mesh-TensorFlow: Deep learning for supercomputers. In *Neural Information Processing Systems*.

Peng Shi and Jimmy Lin. 2019. Simple BERT models for relation extraction and semantic role labeling.

Richard Socher, Milind Ganjoo, Christopher D Manning, and Andrew Ng. 2013. Zero-shot learning through cross-modal transfer. In *Advances in neural information processing systems*, pages 935–943.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. 2019. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In the Proceedings of ICLR.

Adina Williams, Nikita Nangia, and Samuel Bowman. 2018. A broad-coverage challenge corpus for sentence understanding through inference. 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 1112–1122, New Orleans, Louisiana. Association for Computational Linguistics.

Ronald J Williams and David Zipser. 1989. A learning algorithm for continually running fully recurrent neural networks. *Neural computation*, 1(2):270–280.

Wenpeng Yin, Jamaal Hay, and Dan Roth. 2019. Benchmarking zero-shot text classification: Datasets, evaluation and entailment approach. 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 3914–3923, Hong Kong, China. Association for Computational Linguistics.

John M. Zelle and Raymond J. Mooney. 1996. Learning to parse database queries using inductive logic programming. In *AAAI/IAAI, Vol. 2*.

Luke Zettlemoyer and Michael Collins. 2005. Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars. In *UAI*.

Sheng Zhang, Xiaodong Liu, Jingjing Liu, Jianfeng Gao, Kevin Duh, and Benjamin Van Durme. 2018. Record: Bridging the gap between human and machine commonsense reading comprehension. *arXiv preprint arXiv:1810.12885*.

Ben Zhou, Daniel Khashabi, Chen-Tse Tsai, and Dan Roth. 2018. Zero-shot open entity typing as type-compatible grounding. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2065–2076, Brussels, Belgium. Association for Computational Linguistics.

A Gathering Passages

We used Google and Bing search engines to gather documents for our task descriptions, creating a custom endpoint with a limited number of websites (described in Section 4.3) for each domain. Each task description was processed by removing stop words and then used as a query through the respective custom search endpoint. This allowed us to retrieve search snippets and URLs that could be used for further processing.

We used each search snippet to generate the full passage, retrieving the full text of any paragraph that was contained in the snippet, or a random amount (between 0 and 3) of sentences before and after each snippet if the full length of the passage exceeded 300 words. This ensured that we maintained crucial information from the query while mitigating potential bias from the search engine.

B Data Preprocessing

In order to facilitate training a text-to-text model on ZEST, we took each task and generated a collection of input-output text pairs. These pairs were then regarded as individual examples in the training—we did not explore approaches that keep examples grouped together by their task. To generate each input, we prepended "zeroshot question: " before the task description and "zeroshot context: " before the corresponding document. In the case of T5, we appended two newline characters to each and then joined them together, whereas BART used a single space. For each output, we simply used the target ZEST provides.

C Training Details

To facilitate reproducing our experiments, this appendix provides additional details on how we trained and ran predictions for the models. Code to reproduce the baseline results is available from https://allenai.org/data/zest.

C.1 T5 Details

Our baselines build off the T5 11B model (Raffel et al., 2019): a text-to-text encoder-decoder structured transformer pretrained via masked language modeling and multi-tasking. T5 11B has 11 billion parameters. Our training, evaluation, and modeling code used the original implementation released with the T5 work.[7]

[7] https://github.com/google-research/text-to-text-transfer-transformer

Training, Evaluation, & Hyper-parameters Since T5 frames tasks as text-to-text, the model was trained via teacher forcing (Williams and Zipser, 1989). Fixed hyper-parameters include an input sequence length of 512, an output sequence length of 512, and a batch size of 32 examples (i.e., instances of tasks, not tasks themselves—see **Data Preprocessing**). The ZEST + MTL baseline equally weighted each component dataset during training, sampling them at the same rate. To tune the learning rate, for each T5 baseline we performed a grid-search over four values: 5e-4, 1e-3, 2e-3, and 4e-3. The best learning rate for each baseline was 4e-3 for context-only, 1e-3 for question-only, 1e-3 for ZEST-only (full data), and 5e-4 for ZEST + MTL. The model was trained for 25,000 updates with checkpoints taken approximately every 2,500 steps. Throughout training, we kept the 10 most recent checkpoints. All other training specifics were identical to those used in the original T5 work (Raffel et al., 2019). For early stopping, we chose the checkpoints with the highest per-instance accuracy on dev to evaluate on test.[8]

Hardware & Compute We trained the T5 models using three v3-256 TPUs on Google Cloud, using one TPU per model and running experiments in parallel. The T5 implementation we built off integrates with Mesh Tensorflow (Shazeer et al., 2018), which provides automatic data and model parallelism. For training, we set a model parallelism of 16. All T5 baselines trained the same model (T5 11B), only on different data. Training took 2 hours, 44 minutes, and 38 seconds on average with a standard deviation of 15 minutes and 28 seconds across the 16 runs. Evaluation on the validation set for the ZEST-only (full data), context-only, and question-only baselines took on average 26 minutes and 9 seconds with a standard deviation of 1 minute and 4 seconds across 12 runs, while evaluation for the ZEST + MTL baseline took on average 45 minutes and 15 seconds with a standard deviation of 1 minute and 11 seconds across 4 runs.[9]

[8] Note that this metric differs from the evaluation we use for reporting results, which is more complex to compute.

[9] The multi-task baseline took longer to evaluate because we also evaluated it on the other tasks besides ZEST.

C.2 BART Details

BART (Lewis et al., 2020) is a text-to-text encoder-decoder structured transformer pretrained with a denoising autoencoding objective. We used the BART-large model with 406 million parameters as implemented in `transformers`[10]. We followed the original hyperparameters recommended by the authors for fine tuning for summarization[11] with the exception of tuning the batch size, learning rate, number of training epochs, and sequence lengths. In particular, we used a batch size of 32, maximum source/target sequence lengths of 512/64, four beams for decoding, weight decay of 0.01 and 0.1 label smoothing. Learning rate was tuned in [3e-5, 5e-5] and number of epochs in [3, 5, 10, 15, 20], with the best model on the development set having learning rate=3e-5 for 15 epochs. Training took approximately 3.5 minutes per epoch on a single RTX 8000 GPU. Detailed results for the best model are shown in Table 9.

D Evaluation Details

As described in Section 5.1, we evaluate the model in a rigorous manner in order to test how well it truly understands each task. We follow conventions established from previous work in the field (Dua et al., 2019; Dasigi et al., 2019) in evaluating typical Reading Comprehension benchmarks and expand upon them, to account for novel output structures.

Evaluating Classification Classification evaluation is straightforward, taking the modified F1 metric (defined in Section 5.1) of the `yes`, `no`, and `NA` classes.

Evaluating Answer Spans We evaluate extracted answer spans by first aligning the gold and predicted answers (in the case of multiple extracted spans) and computing the F1 word overlap score. We take the max F1 score (with respect to the different answers given by annotators) as the final score for that prediction. This F1 word-overlap score is calculated from the code of (Dua et al., 2019). We then compute the task F1 score following Section 5.1.

Evaluating Output Structure Questions As each output structure answer could contain multiple entities (contained in dictionaries, to use JSON terminology. In Table 1 the entity would be "Bridalveil Fall"), we first align all entities in the predicted and gold answers together. We then use each (key, value) pair as a answer, matching the gold pair to the predicted pair. The score for the value comparisons is evaluated as described in the above two sub-sections, w.r.t whether the value is a classification answer or an extracted answer. We then weight the value score by the key F1 score, as the key is given in the question and is only a reference to the actual answer (e.g. a model should not receive credit for getting the key right, but should receive a penalty for getting the key wrong). Each (key, value) pair in all answers for the given task is used in calculating the final task F1 score, as described in Section 5.1.

E Baseline Results

This appendix provides the full results breakdown for T5 trained on ZEST alone, BART, and human performance. In addition, we've reproduced Table 4 in this appendix for easy comparison.

[10] https://github.com/huggingface/transformers

[11] https://github.com/pytorch/fairseq/blob/master/examples/bart/README.summarization.md

	Dev			Test		
Generalization Type	**Mean**	C@75	C@90	**Mean**	C@75	C@90
Base	71	48	16	63	43	22
Paraphrase	64	36	12	56	32	16
Composition	66	44	22	65	41	15
Semantic Flips	54	27	9	47	18	5
Output Structure	33	20	10	47	10	3
Overall	56	35	14	56	28	12

Table 7: Detailed T5 results for ZEST with multi-task training.

	Dev			Test		
Generalization Type	**Mean**	C@75	C@90	**Mean**	C@75	C@90
Base	69	48	16	62	40	17
Paraphrase	56	28	12	56	33	12
Composition	73	56	22	64	40	15
Semantic Flips	56	27	9	45	15	6
Output Structure	25	0	0	47	14	3
Overall	56	32	12	55	28	11

Table 8: Detailed T5 results for ZEST only training.

	Dev			Test		
Generalization Type	**Mean**	C@75	C@90	**Mean**	C@75	C@90
Base	50	16	8	51	21	7
Paraphrase	39	8	0	41	13	4
Composition	44	15	7	44	13	5
Semantic Flips	42	5	5	34	7	2
Output Structure	23	20	20	19	2	2
Overall	40	13	8	38	11	4

Table 9: Detailed BART-large results for ZEST only training.

	Test		
Generalization Type	**Mean**	C@75	C@90
Base	88	85	54
Paraphrase	85	75	50
Composition	79	67	44
Semantic Flips	68	50	40
Output Structure	51	30	20
Overall	74	61	42

Table 10: Detailed human performance on ZEST.