Unnatural Language Processing: Bridging the Gap Between Synthetic and Natural Language Data

Alana Marzoev [1] **Samuel Madden** [1] **M. Frans Kaashoek** [1] **Michael Cafarella** [2] **Jacob Andreas** [1]

Abstract

Large, human-annotated datasets are central to the development of natural language processing models. Collecting these datasets can be the most challenging part of the development process. We address this problem by introducing a general-purpose technique for "simulation-to-real" transfer in language understanding problems with a delimited set of target behaviors, making it possible to develop models that can interpret natural utterances without natural training data.

We begin with a synthetic data generation procedure, and train a model that can accurately interpret utterances produced by the data generator. To generalize to natural utterances, we automatically find *projections* of natural language utterances onto the support of the synthetic language, using learned sentence embeddings to define a distance metric. With only synthetic training data, our approach matches or outperforms state-of-the-art models trained on natural language data in several domains. These results suggest that simulation-to-real transfer is a practical framework for developing NLP applications, and that improved models for transfer might provide wide-ranging improvements in downstream tasks.

1. Introduction

Data collection remains a major obstacle to the development of learned models for new language processing applications. Large text corpora are available for learning tasks like language modeling and machine translation (Callison-Burch et al., 2011; Chelba et al., 2013). But other classes of NLP models—especially those that interact with the outside world, whether via API calls (e.g., for question answering) or physical actuators (e.g., for personal robotics)—require custom datasets that capture both the full scope of desired behavior in addition to the possible variation in human language. Collecting these large, human-annotated training sets can be an expensive and time-consuming undertaking (Zelle, 1995). In domains governed by well-defined mathematical models, such as physics simulators and graphics engines, one solution to the data scarcity problem problem is "simulation-to-real" transfer (Tzeng et al., 2016). In sim-to-real approaches, knowledge gained in a simulated environment is later applied in the real world with the ultimate aim of generalizing despite discrepancies between the simulated environment and reality.

In this paper, we explore sim-to-real transfer for natural language processing. We use simple, high-precision grammars as "simulators" to generate synthetic training data for question answering and instruction following problems. While synthetic data generation provides potentially unlimited supervision for the learning of these behaviors, interpretation of synthetic utterances may itself constitute a challenging machine learning problem when the desired outputs require nontrivial inference for parsing, planning or perception (Luketina et al., 2019). Given a model with high accuracy on the synthetic training distribution, we interpret natural user utterances from outside this distribution by mapping each natural utterance to a synthetic one and interpreting the synthetic utterance with the learned model. Using pretrained sentence embeddings (Devlin et al., 2018), we define an (approximately) meaning-preserving projection operation from the set of all sentences to those the model has been trained to interpret. Together, labeled synthetic utterances and unsupervised representation learning enable generalization to real language.

Through experiments, we demonstrate the effectiveness of sim-to-real transfer on a variety of domains. On a suite of eight semantic parsing datasets (Wang et al., 2015), sim-to-real matches the performance of the best *supervised* semantic parser on three of eight tasks using no natural training data. On a grounded instruction following benchmark involving challenging navigation in a gridworld environment (Chevalier-Boisvert et al., 2018), our approach to sim-to-real transfer again surpasses the performance of a standard model fine-tuned with human annotations promise of sim-to-real as a development paradigm for natural lan-

[1]Massachusetts Institute of Technology [2]University of Michigan. Correspondence to: Alana Marzoev <marzoev@mit.edu>.

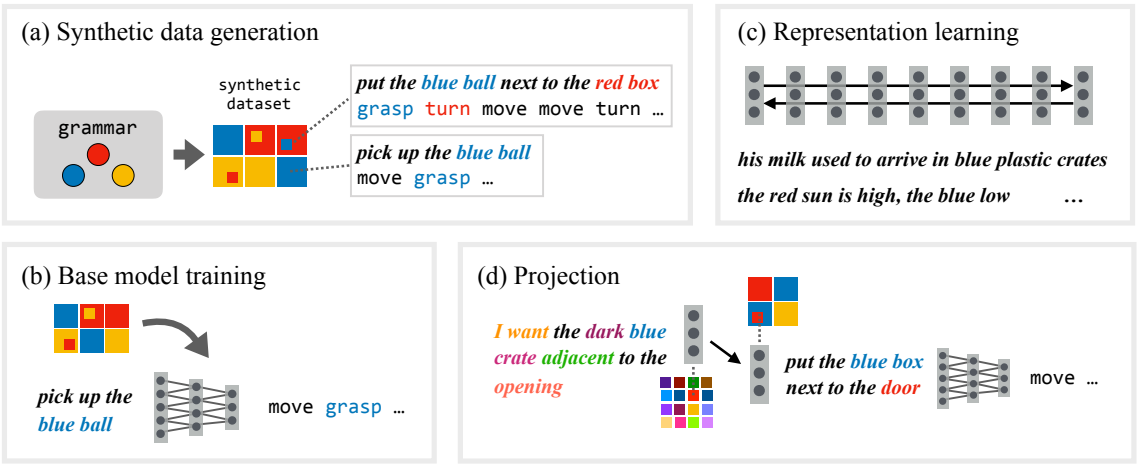


Figure 1. Overview of our approach to synthetic-to-real transfer in language understanding tasks. (a) Training examples are generated from a synthetic data generation procedure that covers desired model behaviors but a limited range of linguistic variation. (b) This data is used to train a model that can correctly interpret synthetic utterances. (c) Separately, sentence representations are learned using a masked language modeling scheme like BERT. (d) To interpret human-generated model inputs from a broader distribution, we first *project* onto the set of sentences reachable by the synthetic data generation procedure, and then interpret the projected sentence with the trained model.

guage processing applications. By leveraging two cheap, readily-available sources of supervision—unaligned natural language text and synthetic language about target tasks—it is possible to build broad-coverage systems for language understanding without a labor-intensive annotation process. Improved models for sim-to-real could lead to positive and wide-ranging effects on downstream tasks and applications. To encourage progress on models for transfer and to further reduce the developer effort associated with building new grounded language understanding, we release (1) a set of new human annotations for a popular policy learning benchmark with synthetic instructions and (2) code implementing the sentence-projection operation. [1]

2. Grammar engineering for grounded language learning

Sim-to-real transfer requires to a simulator: an automated procedure that can generate labeled input–output examples. The experiments in this paper focus on two *language understanding* problems, question answering and instruction following. Both problems require mapping natural language inputs (e.g., *go to the red door*) to concrete interpretations (e.g., a program `(go-to (location red_door))`, which may be executed to produce a sequence of situated low-level actions (`move, turn_r, move, move`)). Many different sentences may be associated with the same program (*find a red door*, *navigate to the red door*, etc.).

[1]Both are available for download at https://github.com/unnatural-language/sim2real.

A simulator for situated language understanding can be implemented as an injective expert-designed *inverse* function, possibly multi-valued, mapping each program (e.g., `(go-to (location yellow_door))`) to a set of distinct input sentences that induce the program (e.g., "go to the location of the yellow door"). This inverse function defines a *synthetic data generation procedure* for the task, a direct mapping between canonical programs and plausible input sentences which can be used to generate training data. Mature tools exist for designing such mappings, generally by adapting domain-general grammars (Bender et al., 2015). Synthetic grammars can be engineered to provide full coverage of the learner's *output space*, generating a plausible input sentence for any target program. Synthetic grammars that define injective maps from interpretations to sentences are unambiguous in the sense that any produced natural language sentence corresponds to exactly one program.

While engineered grammars make it possible to generate large amounts labeled training data, they can provide at best limited coverage of natural language (Erbach & Uszkoreit, 1990): each program is mapped to a small number of plausible input sentences (as defined by the expert-written inverse function), meaning that the full spectrum of linguistic variation is not observed in the generated data. This leads to catastrophic distributional shifts for models trained exclusively on this generated synthetic data: for example, on a drone instruction following benchmark, Blukis et al. report a test-time accuracy gap of over 54% between models trained on synthetic data and those trained on real user utterances (2018). In the next section, we describe a simple

modeling approach that allows system-builders to use synthetic data to train models that are robust to natural inputs.

3. Sim-to-real transfer for NLP

Using data from a synthetic grammar as described in Section 2, we can train a model such that for any desired output, there is *some* input sentence that produces that prediction. The behavior of this model will be undetermined on most inputs. However, to interpret an out-of-scope utterance x using a model trained on synthetic data, it is sufficient to find some synthetic $\tilde{x}$ with the same meaning as x, and ensure that the model's prediction is the same for x and $\tilde{x}$. In other words, all that is required for sim-to-real transfer is a model of *paraphrase* relations and a synthetic data distribution rich enough to contain a paraphrase of every task-relevant input.

In this framework, one possible approach to the sim-to-real problem would involve building a *paraphrase generation model* that could generate natural paraphrases of synthetic sentence, then training the language understanding model on a dataset augmented with paraphrases (Basik et al., 2018; Su & Yan, 2017). However, collecting training data for such a paraphrase model might be nearly as challenging as collecting task-specific annotations directly. Instead, we propose to find *synthetic* paraphrases of *natural* sentences, a process which requires only a model of sentence similarity and no supervised paraphrase data at all.

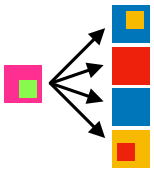
Formally, we wish to produce a **wide-coverage model** $f : \mathcal{X} \to \mathcal{Y}$, where $\mathcal{X}$ is the space of of natural language inputs (e.g., questions or instructions) and $\mathcal{Y}$ is the space of outputs (e.g., meaning representations, action sequences, or dialogue responses). We accomplish this by defining a space of synthetic sentences, $\widetilde{\mathcal{X}}$. and train a **synthetic model** $\tilde{f} : \widetilde{\mathcal{X}} \to \mathcal{Y}$ on an arbitrarily large dataset of synthetic training examples $\widetilde{\mathcal{D}} = \{(\tilde{x}_i, \tilde{y}_i)\}$. To generalize to real sentences, all that is necessary is a function $\pi : \mathcal{X} \to \widetilde{\mathcal{X}}$ that "projects" real sentences onto their synthetic paraphrases. We then define:

$$f(x) = \tilde{f}(\pi(x)) \; . \tag{1}$$

The synthetic model $\tilde{f}$ can be trained using standard machine learning techniques as appropriate for the task. It remains only to define the projection function π. Inspired by recent advances in language modeling and representation learning, we choose to use pretrained sentence representations as the basis for this projection function, mapping from natural language to synthetic utterances based on similarity in embedding space, under the assumption that rich contextual representations of language can be used to cope with distributional differences between natural and synthetic language.

In the remainder of section, we describe the steps needed to construct a working implementation of the projection function π for language understanding problems of increasing complexity.

3.1. Paraphrase via similarity search



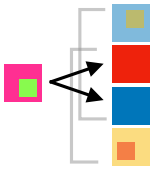
Many self-supervised pretraining schemes for NLP produce vector representations of sentences in which distance in vector space closely tracks semantic similarity (see e.g. recent results on the semantic textual similarity benchmark; Agirre et al., 2016). With this in mind, we propose to project from the natural data distribution $\mathcal{D}$ onto the set of synthetic sentences $\widetilde{\mathcal{X}}$ with respect to a distance function δ defined by a pretrained sentence embedding model $\texttt{embed} : \mathcal{X} \to \mathbb{R}^d$. That is, given a natural language input x, we define:

$$\pi_{\text{full}}(x) = \arg\min_{\tilde{x} \in \widetilde{\mathcal{X}}} \; \delta(\texttt{embed}(x), \texttt{embed}(\tilde{x})) \tag{2}$$

and finally predict $f(x) = \tilde{f}(\pi_{\text{full}}(x))$ as in Equation 1. This framework is agnostic to choice of embedding model and distance metric. For all experiments in this paper, `embed` returns the average of contextual word representations produced by the `bert-base-uncased` model (Devlin et al., 2018), and $\delta(u, v)$ is the cosine distance $1 - u^\top v / (\|u\| \|v\|)$ (or the variant described in Section 3.4).

This projection method is straightforward but computationally expensive: each projection requires enumerating and embedding every utterance that can be generated by the synthetic grammar. Depending on the problem domain, the size of $\widetilde{\mathcal{X}}$ can vary significantly. For simpler problems, there may be hundreds to thousands of examples. For complex problems; $|\widetilde{\mathcal{X}}|$ might be larger or even infinite, making explicit enumeration intractable or impossible.

3.2. Amortized inference with locality sensitive hashing



To reduce the $O(n)$ search cost of the $\arg\min$ in Equation 2, we use *locality sensitive hashing* (LSH; Gionis et al., 1999) to reduce the search space. Rather than requiring a search across every candidate synthetic sentence for each new natural language input, locality sensitive hashing allows for a one-time $O(|\widetilde{\mathcal{X}}|)$ preprocessing step to compute hashes for synthetic sentences, such that sentences with similar embeddings fall into nearby buckets. We then need search over only a constant number of nearby buckets of any given input natural language sentence to find all candidate synthetic sentences.

To implement locality sensitive hashing we use Simhash (Charikar, 2002), an LSH technique based on random projections of vectors. Simhash takes as

input a high dimensional vector in $\mathbb{R}^d$ and outputs an f-dimensional vectors of bits, called a *fingerprint*, with the property that similar input vectors with regards to cosine similarity generate similar fingerprints. To accomplish this, Simhash generates f random hyperplanes in d-dimensional space (denoted $\ell_1 \cdots \ell_f$), computes the dot product of each hyperplane with the input vector, and outputs an f-bit hash corresponding to the sign of each dot product:

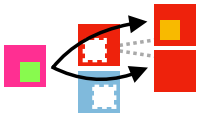
$$h(x) = [\text{sgn}(\ell_1^\top \texttt{embed}(x)),\ \cdots,\ \text{sgn}(\ell_f^\top \texttt{embed}(x))] \quad (3)$$

Since nearby points tend to lie on the same side of random hyperplanes, the probability that two points have the same fingerprint is proportional to the angle between them, and thus directly related to their cosine similarity (Charikar, 2002). By bucketing datapoints according to their fingerprints, the exhaustive search in Equation 2 can be restricted to those points with the same signature as x:

$$\pi_{\text{lsh}}(x) = \underset{\tilde{x} \in \widetilde{\mathcal{X}}:\ h(\tilde{x}) = h(x)}{\arg\min} \delta(\texttt{embed}(x), \texttt{embed}(\tilde{x})) \quad (4)$$

Constructing a data structure to support LSH still requires exhaustively enumerating the full set of candidate synthetic utterances once, but reduces each subsequent lookup to time proportional to bucket size (controlled by f), which can be tuned to trade off between speed and recall.

3.3. Fast inference with hierarchical projection



To further optimize the search process of the $\arg\min$ within π, and to extend the projection techniques to domains where even a single enumeration of $\widetilde{\mathcal{X}}$ is intractable, we introduce a *hierarchical* procedure for computing π. Rather than construct the LSH search data structure ahead of time, we construct a subset of it dynamically for each input query. This procedure is not possible for general nearest neighbor search problems, but here we can rely on special structure: synthetic utterances are generated from a grammar, and the `embed` function can be trained to provide a meaningful measure of similarity *even for incomplete derivations under the grammar*.

To perform hierarchical search, we assume that our synthetic data is generated from a context-free grammar (Hopcroft et al., 2001). For example, the example sentence *pick up the red ball* may be generated by the sequence of derivation steps `$root` $\rightarrow$ *pick up the* `$item` $\rightarrow$ *pick up the ball* (arrows $\rightarrow$ denote reachability in one step and `$dollar` signs denote nonterminal symbols).

We use this derivation process to compute the $\arg\min$ from previous equations iteratively, by repeatedly selecting a nonterminal to instantiate, instantiating it, and repeating the process until a complete synthetic sentence is generated. We illustrate our approach through an example:

Given an input natural language utterance, *I want the dark blue crate adjacent to the opening* (Figure 1), we first rank expansions of the root symbol of the synthetic grammar. Here we rely specifically on the use of a masked language model for sentence representations: we replace each nonterminal symbol with a `[MASK]` token to obtain meaningful similarity between complete real sentences and partially instantiated synthetic ones.

Letting $\widetilde{\mathcal{X}}(\texttt{\$root})$ denote the set of (complete or incomplete) sentences that can be derived from the incomplete derivation `$root` in a single step, we have:

$$\pi^{(1)}_{\text{hier}}(x) = \underset{\tilde{x} \in \widetilde{\mathcal{D}}(\texttt{\$root})}{\arg\min} \delta(\texttt{embed}(x), \texttt{embed}(\tilde{x})) \quad (5)$$

as before. For the example in Figure 2, $\pi^{(1)}_{\text{hier}}(x) =$ *put the* `$item` *next to* `$item`.

The process is then repeated for all unexpanded nonterminals in the partial derivation until a complete sentence is generated. This procedure corresponds to greedy search over sentences generated by the CFG, with measured similarity between partial derivations and the target string x as a search heuristic. This procedure can be straightforwardly extended to a beam search by computing the k lowest-scoring expansions rather than just the $\arg\min$ at each step.

If the grammar has a high branching factor (i.e. the number of expansions for each nonterminal symbol is large), it may also be useful to incorporate other search heuristics. For all experiments in this paper, we restrict the set of expansions for noun phrase nonterminals. We begin by using a pretrained chunker (Sang & Buchholz, 2000) to label noun phrases in the natural language input x (in Figure 2 running example this gives *the dark blue crate* and *the opening*). When expanding noun phrase nonterminals, we first align the nonterminal to a noun chunk in x based on similarity

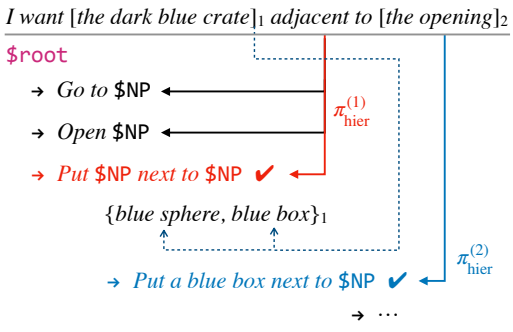


Figure 2. Hierarchical projection. Given a natural language input, we search for a high-scoring utterance generated by a fixed CFG, using similarity between sentences and partial derivations as a search heuristic. An additional heuristic scores noun phrases locally by measuring their similarity with noun chunks extracted from the input sentence.

between the chunk and all right-hand sides for the nonterminal, and finally select the single right-hand side that is most similar to the aligned noun chunk. Greedy population of noun phrases ensures greater beam diversity of sentence-level templates in beam hypotheses. We additionally discard any hypotheses in which there is a mismatch between the number of noun phrases in the query sentence x and the number of groups of adjacent nonterminals in the top-level partial derivation $\widetilde{\mathcal{D}}^{(1)}$. For example, if the template calls for two distinct entities, but the natural language utterance only contains one (e.g. *go to the green door*), then the derivation is immediately discarded.

3.4. Tunable models with matching scores

Hierarchical representations of synthetic utterances makes it possible to improve scoring as well as search. Consider the set of candidates:

1. *go through the yellow door and pick up the red ball*
2. *go through the red door and pick up the yellow ball*

and a natural language input referencing a *red ball* and yellow door. Candidate (1) should be prioritized over candidate (2), but in early experiments we found that fine-grained attribute–value distinctions are not always captured by top-level sentence embedding similarities. To improve this, we also experiment with a modified version of the similarity measure δ that incorporates fine-grained lexical similarity in addition to sentence similarity.

To compute this new distance function δ', we model the problem as one in finding a minimum weight matching in bipartite graphs. Suppose we have a set of (complete) candidate synthetic utterances derived from either a flat or hierarchical projection procedure as defined above. As above, we extract noun chunks from the natural language query and each synthetic utterance, representing each chunk as a node in a graph connected to each node from the utterance with edge weights equivalent to the distance between the respective *phrases* in embedding space. We use the Hungarian algorithm (Kuhn, 1955) to solve this matching problem, and incorporate the computed minimum syntactic cost δ_{sub} from each candidate sentence into the overall distance function δ. In particular, we redefine the similarity function as

$$\begin{aligned}\delta'(x, \tilde{x}) =& \delta(\mathsf{embed}(x), \mathsf{embed}(\tilde{x})) \\ &+ \alpha \sum_{(x', \tilde{x}') \in M} \delta(\mathsf{embed}(x'), \mathsf{embed}(\tilde{x}') \end{aligned} \quad (6)$$

where M is the matching found by the Hungarian algorithm and α is a hyperparameter that can be tuned as required based on the grammar of the problem domain.

This scoring strategy can have detrimental effect if α is set too high — one example of a where δ_{sub} incorrectly overpowers δ is when "go through the yellow door and pick up the red ball" is projected onto an utterance containing those extra details that is not semantically equivalent, such as "put the red ball next to the yellow door". However, in early experiments on BABYAI dataset we found the additional fine-grained control provided by matching scores led to slight improvements in projection accuracy. BABYAI experiments use an α of 0.1.

4. Evaluation

We evaluate our approach on two tasks with markedly different output spaces and data requirements: the OVERNIGHT semantic parsing benchmark (Wang et al., 2015) and the BABYAI instruction following benchmark (Chevalier-Boisvert et al., 2018). These experiments aim (1) to test the flexibility of the general sim-to-real framework across problems with complex compositionality and complex interaction, and (2) to measure the effectiveness of projection-based sentence interpretation relative to other ways of learning from pretrained representations and synthetic data.

Both tasks come with predefined synthetic data generators intended for uses other than our projection procedure. The effectiveness of the approach described in this paper inevitably depends on the quality of the data synthesizer; by demonstrating good performance using off-the-shelf synthesizers from previous work, we hope to demonstrate the generality and robustness of the proposed approach. We expect that better results could be obtained using a grammar optimized for peformance; we leave this for future work.

4.1. Models

Experiments on both benchmarks use the following models:

seq2seq A baseline neural sequence model. For the semantic parsing task, this is a standard LSTM encoder–decoder architecture with attention (Bahdanau et al., 2015). (Initial experiments indicated that this model significantly outperformed the dependency-based semantic parsing approach of Wang et al.) We use a 256-dimensional embedding layer and a 1024-dimensional hidden state. For the instruction following task, we reuse the agent implementation provided with the BABYAI dataset, which jointly encodes instructions and environments states with a FiLM module (Perez et al., 2018), and generates sequences of actions with an LSTM decoder. More details are provided in the original work of Chevalier-Boisvert et al. (2018). The seq2seq baseline measures the extent to which available synthetic data in each domain is already good enough to obtain sim-to-real transfer from standard models without specialization.

seq2seq+BERT As discussed in Artetxe & Schwenk (2019), the standard approach to the sort of zero-shot transfer investigated here is to train models of the kind described above not on raw sequences of input tokens, but rather on sequences of contextual embeddings provided by a pretrained representation model. The intuition is that these representations capture enough language-general information—e.g. about which words are similar and which syntactic distinctions are meaningful—that a learned model of the relationship between input representations and task predictions in the source (synthetic) domain will carry over to input representations in the target domain.

Concretely, seq2seq+BERT models are constructed for both tasks by taking the language encoder and replacing its learned word embedding layer with contextual embeddings provided by the `bert-base-uncased` model of Devlin et al.. This is analogous to the strategy employed by Lindemann et al. (2019) and has been shown to be an effective way to incorporate pretraining into a variety of semantic prediction tasks. Though the bulk of this paper focuses on the projection technique, the experiments we present are also, to the best of our knowledge, the first to systematically evaluate even this basic pretraining and transfer scheme in the context of synthetic data.

projection The final model we evaluate is our projection approach described in Section 3. As discussed above, the projection model is built from the same pieces as the baselines: it relies on the base seq2seq model for interpreting synthetic data, the similarity function induced by BERT embeddings in the projection step.

4.2. Experiments

Semantic parsing In semantic parsing, the goal is to learn a mapping from natural language utterances to formal, executable representations of meaning (Figure 3). These meaning representations can then be used to as inputs to database engines or formal reasoning systems. The OVERNIGHT benchmark consists of eight semantic parsing datasets covering a range of semantic phenomena, and is designed to test efficient learning from small numbers of examples—individual datasets range in size from 640–3535 examples.

The semantic originally described by Wang et al. for this task was equipped with a "canonical grammar"—a compact set of rules describing a mapping from logical forms to (somewhat stilted) natural language strings (*meeting whose end time is smaller than start time of weekly standup*). In the original work, this grammar was used as a source of features for reranking logical forms. Here we instead use it as a source of *synthetic training data*, enumerating strings from the grammar paired with logical forms, and using these pairs to train the base seq2seq model. The dataset also comes with a set of natural language annotations on train- and test-set logical forms.

real	*show me the meeting starting latest in the day*
synth	*meeting that has the largest start time*
LF	`(max meeting (get start_time))`
real	what recipes posting date is at least the same as rice pudding
synth	recipe whose posting date is at least posting date of rice pudding
LF	`(filter recipe (≥ (get posting-date) ((get posting-date) RICE-PUDDING)))`

Figure 3. Example sentences from the *calendar* and *recipe* OVERNIGHT domains. *real* is a human-generated utterance, *synth* is a synthetic utterance from the domain grammar of TODO, and *LF* is the target logical expression.

Results are shown in Table 1. We report logical form exact match accuracies for the eight OVERNIGHT datasets.[2] Three data conditions are evaluated: *synth*, which uses synthetic-only training data, *real*, which uses only human-annotated training data, and *both*, which combines the two. Human annotations are used for the test evaluation in all data conditions. For the *both* condition, we found that simply concatenating the two datasets was reliably better than any fine-tuning scheme.

It can be seen that the projection-based approach to learning from synthetic data consistently outperforms sequence-to-sequence models with or without pretrained representations. In fact, projection outperforms the best *supervised* approach in three domains. These results demonstrate that projection is an effective mechanism for sim-to-real transfer in tasks with a challenging *language understanding* component, enabling the use of synthetic strings to bootstrap a model that can perform fine-grained linguistic analysis of real ones.

Instruction following The BABYAI instruction following dataset presents a range of challenging manipulation and navigation tasks in two-dimensional gridworld environments—agents are trained to achieve a wide variety of compositional goals (e.g. *put a yellow box next to a gray door*, Figure 4) specified by sentences from an artificial grammar.

This dataset was originally designed to explore generalization in reinforcement and imitation learning, not natural language understanding, but it provides a flexible and extensible test-bed for studying questions around generalization in natural language as well. We collected a dataset of roughly 5000 natural language utterances by showing Mechanical Turk workers videos of agents executing plans

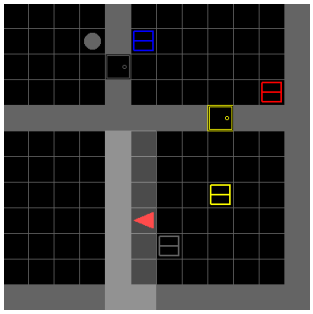
[2]The original work used a coarser-grained denotation match evaluation.

Data	Model	basketball	blocks	calendar	housing	pubs	recipes	restaurants	social	mean
synth	projection	0.47	**0.27**	**0.32**	**0.36**	**0.34**	**0.49**	**0.43**	0.28	**0.37**
	seq2seq	0.27	0.08	0.22	0.16	0.24	0.21	0.23	0.07	0.19
	seq2seq+BERT	**0.60**	0.21	0.31	0.31	**0.34**	0.36	0.36	**0.31**	0.35
real	seq2seq	0.45	0.10	0.27	0.19	0.30	0.38	0.25	0.20	0.27
	seq2seq+BERT	0.63	0.26	0.37	0.29	0.44	0.53	0.43	0.42	0.42
both	seq2seq	0.01	0.25	0.13	0.10	0.52	0.18	0.42	0.36	0.25
	seq2seq+BERT	0.67	0.23	0.43	0.29	0.48	0.29	0.36	0.38	0.39

Table 1. OVERNIGHT semantic parsing logical form accuracies on human-generated questions for models trained with synthetic data, real data, or both. **Bold** values indicate the best-performing model under the *synth* data condition, while underlined values indicate the best-performing model under *any* data condition. The projection approach consistently outperforms other models in the synthetic-only condition, and even manages to outperform the best fully-supervised model in three domains.

Data	Model	success	reward
synth	projection	**0.63**	**0.72**
	seq2seq	0.56	0.67
	seq2seq+BERT	0.54	0.66
both	seq2seq	**0.59**	**0.71**
	seq2seq+BERT	0.53	0.65

Table 2. Instruction following accuracies on the BABYAI `SynthLoc` task with human-generated instructions and models trained on synthetic or a mix of synthetic and real data. **Bold** values indicate the best-performing model for a given data condition and underlined values indicate the best-performing model under any data condition. Projection gives the best results on both success (task completion) and discounted reward metrics, outperforming other models trained on only synthetic data as well as models fine-tuned on 2048 real examples.



synth: *put a yellow box next to a gray door*
real: *drive around and put the yellow block beside the top door*

Figure 4. Example from the BABYAI dataset. Agents are given tasks with language-like specifications, and must execute a sequence of low-level actions in the environment to complete them. We augment this dataset with a set of human instructions, and evaluate generalization of agents trained only on synthetic goal specifications to novel human requests.

in BABYAI environments (specifically from the `SynthLoc` training set). Workers did not see the original BABYAI commands, but were asked to generate new instructions that would cause a robot to accomplish the goal displayed in the video. The dataset of human annotations is released with this paper.

As in the OVERNIGHT experiments, we train (in this case via imitation learning) on either the underlying synthetic instructions alone, or a mix of synthetic and real instructions. We evaluate various models' ability to generalize to a human-annotated test set. Unlike the OVERNIGHT datasets, bootstrapping even a seq2seq model that achieves good performance on a test set with *synthetic* instructions requires enormous amounts of data: the existing synthetic training set contains a million demonstrations. To adapt to human instructions, we fine-tune this baseline model (rather than concatenating training sets as above).

Results are shown in Table 2. Again, projection is the best way to incorporate pretrained representations: introducing BERT embeddings into the underlying sequence-to-sequence model makes performance worse. (We attribute this to overfitting to the limited set of word embeddings observed during training.) Again, as well, projection is effective enough to allow synthetic-only training to outperform fine-tuning on a small amount of human data. These results indicate that sim-to-real transfer is also an effective strategy for tasks whose sample complexity results from the difficulty of an underlying planning problem rather than language understanding as such.

5. Related work

The technique described in this paper brings together three overlapping lines of work in machine learning and natural language processing:

Sim-to-real transfer in robotics The most immediate inspiration for the present work comes from work on general-

ization beyond simulation in robotics and other challenging control problems (Tzeng et al., 2016) Simulators provide convenient environments for training agents because the poor sample efficiency of current policy learning algorithms are mitigated by faster-than-real-time interaction and unlimited data. “Learning” in simulation amounts to amortizing the inference problem associated with optimal control (Levine, 2018): correct behavior is fully determined by the specification of the environment but poses a hard search problem.

However, simulators do not represent reality with complete accuracy, and agents trained to perform specific tasks in simulators often fail to perform these same tasks in the real world. Various techniques have been proposed for bridging this this “reality gap" (Tobin et al., 2017). Language processing applications suffer from similar data availability issues. For grounded language learning problems in particular, “end-to-end” architectures make it hard to disentangle challenging *language learning* problems from challenging *inference* problems of the kind discussed above (Andreas & Klein, 2015). Our approach aims to offload most learning of linguistic representations to pretraining without sacrificing the expressive power of models for either part of the problem.

Representation learning The last two years have seen remarkable success at learning general-purpose representations of language from proxy tasks like masked language modeling (Peters et al., 2018; Devlin et al., 2018). For tasks with limited in-domain training data available, a widespread approach has been to begin with these pretrained representations, train on data from related source domains in which labels are available, and rely on similarity of pretrained representations in source and target domains to achieve transfer (Artetxe & Schwenk, 2019). One surprising finding in the current work is that this strategy is of only limited effectiveness in the specific case of synthetic-to-real transfer: it is better to use these pretrained representations to find paraphrases from the target domain back to the synthetic source domain, and interpret sentences normally after paraphrasing, than it is to rely on transfer of representations themselves within a larger learned model.

Grammar engineering Finally, we view our work as offering a new perspective in a longstanding conversation around the role of grammar engineering and other rule-based approaches in natural language processing (Chiticariu et al., 2013). Rule-based approaches are criticized for being impossible to scale to the full complexity of natural language data, providing good coverage of “standard” phenomena but require prohibitive additional to model the numerous special cases and exceptions that characterize human language (Norvig, 2017).

However, engineered grammars and synthetic data generation procedures also offer a number of advantages. Mature engineering tools and syntactic resources are available for many major languages (Flickinger, 2011). Hand-written grammars are straightforward to implement and can serve as cost-effective tools for language generation in larger data-synthesis pipelines; they also enable developers to quickly incorporate new linguistic phenomena when they are found to be outside model capacity. Moreover, since the relative frequency syntactic and semantic phenomena can be specified by engineers, the poor accuracy on long-tail pheonomena that is observed in models trained on *natural* datasets (Bamman, 2017) can be mitigated by flattening out the relevant distributions in the data generation process.

The approach presented in this paper, and the prospect of more general sim-to-real transfer in NLP, offers to turn the partial solution offered by current grammar engineering approaches into a more complete one: model builders with limited data collection ability can construct high-quality synthetic data distributions with general-purpose language resources, project project onto these distributions with high accuracy, and eventually obtain good end-to-end performance without end-to-end data collection or training.

6. Discussion

We have described a family of techniques for sim-to-real transfer in natural language processing contexts, a comparative analysis of these techniques, a new benchmark for measuring sim-to-real transfer performance, and a practical software framework for bootstrapping synthetic grammars.

The projection procedure described in this paper suggests a new approach to bootstrapping natural language applications. In this paradigm, key insights about the relationship between the world and language are explicitly encoded into the declarative synthetic data generation procedure rather than implicitly in the model’s structure or through the use of a human-annotated dataset, and can take advantage of advances in machine learning and structured knowledge about human language. Developers who want to build applications in new domains can therefore hand-engineer synthetic grammars, use the generated data to train domain-specific machine learning models, and use projections to paraphrase test time natural language examples into their synthetic counterparts. This makes it possible to recover substantial amounts of model accuracy that otherwise would have been lost when switching from the distribution of synthetic to natural language utterances.

References

Agirre, E., Banea, C., Cer, D., Diab, M., Gonzalez Agirre, A., Mihalcea, R., Rigau Claramunt, G., and Wiebe, J.

Semeval-2016 task 1: Semantic textual similarity, monolingual and cross-lingual evaluation. In *SemEval-2016. 10th International Workshop on Semantic Evaluation; 2016 Jun 16-17; San Diego, CA. Stroudsburg (PA): ACL; 2016. p. 497-511.* ACL (Association for Computational Linguistics), 2016.

Andreas, J. and Klein, D. Alignment-based compositional semantics for instruction following. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, 2015.

Artetxe, M. and Schwenk, H. Massively multilingual sentence embeddings for zero-shot cross-lingual transfer and beyond. *Transactions of the Association for Computational Linguistics*, 7:597–610, 2019.

Bahdanau, D., Cho, K., and Bengio, Y. Neural machine translation by jointly learning to align and translate. In *Proceedings of the International Conference on Learning Representations*, 2015.

Bamman, D. Natural language processing for the long tail. In *DH*, 2017.

Basik, F., Hattasch, B., Ilkhechi, A. R., Usta, A., Ramaswamy, S., Utama, P., Weir, N., Binnig, C., and Çetintemel, U. Dbpal: A learned NL-interface for databases. In *Proceedings of the 2018 International Conference on Management of Data*, pp. 1765–1768, Houston, TX, USA, 2018.

Bender, E. M., Levin, L., Müller, S., Parmentier, Y., and Ranta, A. Proceedings of the grammar engineering across frameworks (GEAF) 2015 workshop. In *Proceedings of the Grammar Engineering Across Frameworks (GEAF) 2015 Workshop*, 2015.

Blukis, V., Misra, D., Knepper, R. A., and Artzi, Y. Mapping navigation instructions to continuous control actions with position-visitation prediction. *arXiv preprint arXiv:1811.04179*, 2018.

Callison-Burch, C., Koehn, P., Monz, C., and Zaidan, O. F. Findings of the 2011 Workshop on Statistical Machine Translation. In *WMT*. Association for Computational Linguistics, 2011.

Charikar, M. S. Similarity estimation techniques from rounding algorithms. In *Proceedings of the Thiry-Fourth Annual ACM Symposium on Theory of Computing*, STOC '02, pp. 380–388, New York, NY, USA, 2002. Association for Computing Machinery. ISBN 1581134959. doi: 10.1145/509907.509965. URL https://doi.org/10.1145/509907.509965.

Chelba, C., Mikolov, T., Schuster, M., Ge, Q., Brants, T., Koehn, P., and Robinson, T. One billion word benchmark for measuring progress in statistical language modeling. *arXiv preprint arXiv:1312.3005*, 2013.

Chevalier-Boisvert, M., Bahdanau, D., Lahlou, S., Willems, L., Saharia, C., Nguyen, T. H., and Bengio, Y. BabyAI: A platform to study the sample efficiency of grounded language learning. In *International Conference on Learning Representations*, 2018.

Chiticariu, L., Li, Y., and Reiss, F. Rule-based information extraction is dead! Long live rule-based information extraction systems! In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pp. 827–832, 2013.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. BERT: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

Erbach, G. and Uszkoreit, H. Grammar engineering: Problems and prospects. *CLAUS report*, 1, 1990.

Flickinger, D. Accuracy vs. robustness in grammar engineering. *Language from a cognitive perspective: Grammar, usage, and processing*, 201:31–50, 2011.

Gionis, A., Indyk, P., Motwani, R., et al. Similarity search in high dimensions via hashing. In *VLDB*, volume 99, pp. 518–529, 1999.

Hopcroft, J. E., Motwani, R., and Ullman, J. D. Introduction to automata theory, languages, and computation. *Acm Sigact News*, 32(1):60–65, 2001.

Kuhn, H. W. The Hungarian method for the assignment problem. *Naval Research Logistics Quarterly*, 2(1-2):83–97, 1955. doi: 10.1002/nav.3800020109. URL https://onlinelibrary.wiley.com/doi/abs/10.1002/nav.3800020109.

Levine, S. Reinforcement learning and control as probabilistic inference: Tutorial and review. *arXiv preprint arXiv:1805.00909*, 2018.

Lindemann, M., Groschwitz, J., and Koller, A. Compositional semantic parsing across graphbanks. *arXiv preprint arXiv:1906.11746*, 2019.

Luketina, J., Nardelli, N., Farquhar, G., Foerster, J., Andreas, J., Grefenstette, E., Whiteson, S., and Rocktäschel, T. A survey of reinforcement learning informed by natural language. In *International Joint Conference on Artificial Intelligence*, 2019.

Norvig, P. On Chomsky and the two cultures of statistical learning. In *Berechenbarkeit der Welt?*, pp. 61–83. Springer, 2017.

Perez, E., Strub, F., De Vries, H., Dumoulin, V., and Courville, A. FiLM: Visual reasoning with a general conditioning layer. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

Peters, M. E., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., and Zettlemoyer, L. Deep contextualized word representations. In *Proceedings of the Annual Meeting of the North American Chapter of the Association for Computational Linguistics*, 2018.

Sang, E. F. and Buchholz, S. Introduction to the CoNLL-2000 shared task: Chunking. *arXiv preprint cs/0009008*, 2000.

Su, Y. and Yan, X. Cross-domain semantic parsing via paraphrasing. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pp. 1235–1246, Copenhagen, Denmark, 2017.

Tobin, J., Fong, R., Ray, A., Schneider, J., Zaremba, W., and Abbeel, P. Domain randomization for transferring deep neural networks from simulation to the real world. In *2017 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, pp. 23–30. IEEE, 2017.

Tzeng, E., Devin, C., Hoffman, J., Finn, C., Peng, X., Levine, S., Saenko, K., and Darrell, T. Towards adapting deep visuomotor representations from simulated to real environments. In *WAFR*, 2016.

Wang, Y., Berant, J., and Liang, P. Building a semantic parser overnight. In *ACL*, 2015.

Zelle, J. *Using Inductive Logic Programming to Automate the Construction of Natural Language Parsers*. PhD thesis, Department of Computer Sciences, The University of Texas at Austin, 1995.