Who's on First?: Probing the Learning and Representation Capabilities of Language Models on Deterministic Closed Domains

David Demeter
Northwestern University
ddemeter@u.northwestern.edu

Doug Downey
Allen Institute for AI
dougd@allenai.org

Abstract

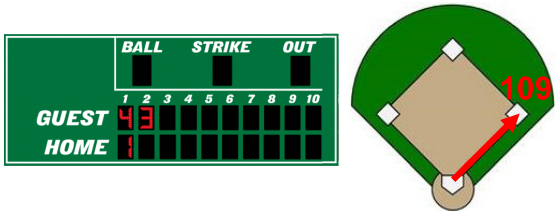
The capabilities of today's natural language processing systems are typically evaluated using large datasets of curated questions and answers. While these are critical benchmarks of progress, they also suffer from weakness due to artificial distributions and incomplete knowledge. Artifacts arising from artificial distributions can overstate language model performance, while incomplete knowledge limits fine-grained analysis.

In this work, we introduce a complementary benchmarking approach based on *SimPlified Language Activity Traces* (SPLAT). SPLATs are corpora of language encodings of activity in some closed domain (we study traces from chess and baseball games in this work). SPLAT datasets use naturally-arising distributions, allow the generation of question-answer pairs at scale, and afford complete knowledge in their closed domains. We show that language models of three different architectures can answer questions about world states using only verb-like encodings of activity. Our approach is extensible to new language models and additional question-answering tasks.

1 Introduction

Language models (LMs) are pre-trained on running text and have been shown to have a remarkable ability to transform this raw observational data into models that capture syntax, semantics, and certain forms of world knowledge such as common sense (Tenney et al., 2019; Rogers et al., 2020). However, accurately measuring the models' representation and learning capabilities can be difficult because the tasks that they are evaluated on can be subjective, require expensive annotation, and suffer from dataset artifacts that allow models to achieve spurious high performance.

In this paper, we propose novel, complementary benchmarks for evaluating the potential of different language model architectures. Our approach involves pre-training and evaluating models on what we term *SimPlified Language Activity Traces* (SPLAT): corpora of language encodings of activity in a closed domain. SPLATs have three characteristics that make them valuable for probing the capabilities of LMs: (a) they are comprised of data from existing, organic distributions *not* designed by the experimenters, helping to prevent spurious artifacts, (b) they can be acquired at scale, enabling the construction of well-trained models,



(a) Tradition Baseball Representation

Balls	Strikes	Outs	Inning	Half	Away	Home	Plate	First	Second	Third	
0	0	0	2	B	7	1	9	_	_	_	<newbatter> (id_109)
1	0	0	2	B	7	1	9	_	_	_	<ball>
2	0	0	2	B	7	1	9	_	_	_	<ball>
2	1	0	2	B	7	1	9	_	_	_	<strike>
0	0	0	2	B	7	1	9	_	_	_	<hit>
0	0	0	2	B	7	1	_	9	_	_	<advance> (plate) (first)

(b) World-State Representation and SPLAT Encoding

```
[start] <sog> ... <newbatter> (id_109)
<ball> <ball> <strike> <hit> <advance>
(plate)(first) [sep] Q1 (first) (id_109)
[end]
```

(c) SPLAT Encoding of "Who's On First?"

Figure 1: **Baseball Illustration.** A typical presentation of baseball is a snapshot of the scoreboard and field (a). We represent these as a sequence of world-states and encode gameplay as a sequence of SPLAT tokens that map to deterministic changes in world-states (b). SPLAT encoding makes it possible to pose question-answer pairs with complete knowledge in a language modeling setting, where a single indicator represents each question (c).

and (c) they admit automatic construction of questions with known, objectively correct answers, that probe the linguistic or cognitive potential of the models.

We introduce two SPLAT datasets based on language-like traces of chess and baseball games (see Figure 1). The tokens in the datasets correspond to actions in the domain. For example, moving a knight from square g1 to square f3 is encoded as the SPLAT token g1f3 (see Appendix A). A sequence of SPLAT tokens can represent entire games in each domain.

We pre-train models for each domain on corpora of raw game traces, and then probe the models' capabilities by fine-tuning on synthetic question-answering data sets. The mechanics of chess and baseball can be captured by simple symbolic programs, meaning that, in contrast to real natural language text, SPLATs afford complete knowledge—that is, it is possible to unambiguously determine the relevant world (game) state at any point in the activity trace. We show how this enables us to create questions that probe key model capabilities, including the ability to construct and maintain world states, learn the rules that govern state transitions, synthesize across states (to determine who is winning), and generate multi-step paths of states to reach a given goal. Further, due to the longstanding popularity of chess and baseball, traces for pretraining and fine-tuning can be obtained at scale.[1]

Many natural language processing (NLP) benchmarks are built from large data sets of human-annotated natural language questions. These data sets often contain artifacts (Gururangan et al., 2018), can be subjective, and are expensive to scale. Prominent examples include GLUE (Wang et al., 2018), decaNLP (McCann et al., 2018), SQuAD 2.0 (Rajpurkar et al., 2018) and CoQA (Reddy et al., 2019), among many others. A number of other benchmarks use natural or artificial language in *synthetic* environments with well-defined underlying mechanisms (Weston et al., 2016; Lake and Baroni, 2018; Hupkes and Zuidema, 2018; Chrupała and Alishahi, 2019). Tasks formulated on these synthetic datasets can probe complex capabilities like state-tracking, can be generated at scale, and have unambiguous answers—but the mechanisms and their distributional characteristics are designed by the experimenters. SPLATs by contrast have a potential advantage in that they correspond to natural, pre-existing distributions of activity that may result in fewer artifacts. A benchmark which maintains a natural distribution, can be generated at scale, and affords complete knowledge of its domain is desirable. While some existing datasets have a number of these qualities, we present the first benchmarks with all three (see Table 1).

	NATURAL DISTRIB.	GENERATE AT SCALE	COMPLETE KNOWLEDGE
Natural Lang. QA			
Synthetic Lang. QA		✓	✓
Unsupervised LMs	✓	✓	
SPLAT	✓	✓	✓

Table 1: **Comparison of Selected Benchmarks.** Natural Language QA includes benchmarks like GLUE (Wang et al., 2018), decaNLP (McCann et al., 2018), SQuAD 2.0 (Rajpurkar et al., 2018) and CoQA (Reddy et al., 2019), among others. Synthetic Language QA includes benchmarks like bAbI (Weston et al., 2016) and other artificial language-based tasks (Lake and Baroni, 2018; Hupkes and Zuidema, 2018; Chrupała and Alishahi, 2019). Unsupervised LMs include autoregressive and masked language models like GPT-2 (Radford et al., 2019) and T5 (Raffel et al., 2020), among others. SPLAT benchmarks fulfill all of our desiderata, making them complementary to the human-annotated and synthetic datasets used for NLP tasks.

Results on these probing tasks show that language models perform surprisingly well, surpassing random baselines by wide margins in almost all cases. However, the models still fall short of the performance that could be achieved by short, human-authored symbolic programs for each domain, highlighting room for improvements and new architectures in language modeling.

Our contributions are twofold:

- We introduce the SPLAT benchmarks, the first extensible code and datasets[2] where it is possible to generate artifact-free question-answer pairs at scale on closed domains with complete knowledge, and
- We show empirical findings across three different architectures highlighting language models' learning and representation capabilities along four dimensions.

[1] We estimate that we can generate 1.3 billion and 150 million artifact-free question-answer pairs for SPLAT_{Chess} and $\text{SPLAT}_{Baseball}$ datasets, respectively.

[2] github.com/daviddemeter/splatbenchmarks

2 Related Work

Multiple benchmark tasks exist to probe the representation and learning capabilities of language models.

2.1 Natural Language Question Answering

Many benchmarks evaluate a variety of capabilities on large datasets comprised of natural language. GLUE (Wang et al., 2018) is one such prominent example, which consists of nine underlying datasets and measures language model capabilities in determining semantic equivalence, performing natural language inference, and estimating sentiment. Other popular benchmarks include decaNLP, SQuAD 2.0, CoQA (McCann et al., 2018; Rajpurkar et al., 2018; Reddy et al., 2019) and open-domain question answering (Chen et al., 2017). To achieve high performance on any individual task, language models must possess a diverse collection of capabilities such as syntax, lexical semantics, reasoning, knowledge acquisition and language understanding, among others. QA evaluations often conflate these capabilities into a single task, making fine-grained analysis of language models' learning and representation capabilities intractable.

Many of these benchmarks use large quantities of running natural language text, but human annotators craft the specific questions and answers used for evaluation. This risks the introduction of unintended artifacts and limits the scale of these datasets (Gururangan et al., 2018). Further, since the evaluation data consists of hand-labeled natural language, the evaluations can be subjective.

2.2 Synthetic Question Answering

The introduction of benchmarks based upon synthetic datasets addresses some of our concerns about scale, subjective evaluations, and fine-grained analysis. The bAbI dataset (Weston et al., 2016) is one prominent example. bAbI is a text-based adventure game that simulates a world with entities and actions that operate on these entities. Natural language descriptions of and questions about these simulated events are generated using various templates to form question-answer pairs. Although these question-answer pairs can be efficiently generated at scale, the resulting text does not follow a natural distribution, risking artifacts that language models can exploit to overstate performance.

However, bAbI does have the advantage of operating in a closed domain where complete knowledge is possible. There are deterministic outcomes traceable to explicit entities and actions. In other words, complete knowledge enables fine-grained analysis of every answer, which is virtually impossible in open-domain question-answering (Chen et al., 2017) due to the size of the source corpora and the imprecise nature of language.

Another family of benchmarks is used to probe discrete language model capabilities on artificial or pseudo languages. SCAN (Lake and Baroni, 2018) evaluates the semantic compositional capabilities of language models using a highly restricted set of natural language commands which map to a smaller set of artificial action sequences. Other works probe the ability of neural networks to learn compositional and hierarchical semantics (Hupkes and Zuidema, 2018) and to represent symbolic semantics (Chrupała and Alishahi, 2019) on simple languages of nested arithmetic expressions. While bAbI uses artificially-generated but natural-language questions, these benchmarks use completely artificial questions.

The use of artificial questions with known underlying mechanisms allows tasks with objective outcomes to be generated at scale. However, this comes at a cost, as the language does not come from a natural distribution.

2.3 Chess and Baseball Tasks

Most prior work in the chess and baseball domains is concerned with either training agents to play the game (David et al., 2016; Schrittwieser et al., 2020) or predicting outcomes using statistical methods. We seek to do neither. Instead, we train language models to operate over these domains and probe the learning and representation capabilities of language models using question-answer pairs.

The closest work to ours (Toshniwal et al., 2021) also trains language models on the verb-like encoding of chess and focuses on legal moves and world-state tracking. Our work uses more stringent evaluation on both of these tasks and incorporates additional tasks related to the synthesis of information and the ability of language models to make multi-hop decisions, and we also generalize the SPLAT evaluation method to the baseball domain.

3 Methods

In this section we present the methodology and motivation used to design our probing tasks. One of our objectives is to provide an extensible framework for other researchers to evaluate new models and create new probes on SPLAT.

3.1 Commonsense Motivation for Probes

Our probing tasks are motivated, in part, by constructs in commonsense reasoning. A key commonsense reasoning task involves determining which future states are more plausible in the world (Zellers et al., 2018). Our first SPLAT task tests models on distinguishing possible from impossible—i.e. determining which moves are legal from any context, and which are illegal. This motivates our **Legal Verbs** probing task.

Commonsense knowledge graph completion (Malaviya et al., 2020) is another core capability in commonsense reasoning and is characterized by extracting implicit knowledge from running text to form knowledge base triples. Predicting state-space variable assignments is a parallel task in SPLAT. For example, populating a series of triples for chess of the form `(square, IsAt, piece)` is equivalent to constructing and maintaining a representation of the state-space. This motivates our **Variable Assignment** probing task.

The ability to learn how to perform reasoning using implicit knowledge, specifically comparison and multi-hop compositionality (Talmor et al., 2019), provides the basis for our third and fourth probing tasks. An important concept for games like chess and baseball is which player is *winning* at any point in the game, which can be estimated by a piece-value system for chess or simply the relative team scores in baseball. For SPLAT encodings, these quantities are not directly observable and must be derived from activity traces. Separately, determining a sequence of actions that yields a specified goal state requires estimating the net results of composing multiple actions, and searching for a satisfactory action sequence. These considerations motivate our **Who's Winning** and **Goal State** probing tasks.

3.2 Formal Description of SPLAT Encoding

Each chess or baseball game $\mathcal{G}$ consists of a sequence of world-states $S_{0:T}$. The world-state is defined by n domain-specific variables e_n, which are assigned a unique value from a set of k variable-specific assignments $a_k^{(n)}$. For example, the queen Q occupying the square at coordinate $d1$ would be expressed as $e_{d1} = Q$. There are sixty-four variables for chess (e.g. all possible coordinates on an 8 x 8 board) and thirteen possible assignments. We follow common notation use upper-case letters for white pieces, lower-case letters for black pieces and $Empty$ for unoccupied squares. A similar construction applies to baseball. Detailed set definitions are presented in Appendices A and B.

The gameplay for chess and baseball is encoded as a sequence of T verbs $\mathcal{V} = \{v_0, v_1, \ldots v_T\}$. Each verb (and associated arguments) v_t represents a discrete action at time t and has a deterministic impact on a world-state S_t. There exists a symbolic program $f()$ which maps verbs v_t to deterministic changes in the world-state, such that:

$$S_t = f(v_t, S_{t-1}) \tag{1}$$

by making assignments $a_k^{(n)}$ for one or more variables e_n.

Consider a baseball example in which player 109 advances from the plate to first base (see Figure 1). The verb encoding v_t for this event is:

$$\langle advance \rangle\ (plate)\ (first)$$

The player advancing from the plate to first is known from the prior world-state S_{t-1}. Using the symbolic program to execute these actions yields world-state S_t:

	Balls	Strikes	Outs	Inning	Half	Away	Home	Plate	First	Second	Third	
S_{t-1}	0	0	0	2	B	7	1	9	_	_	_	
S_t	0	0	0	2	B	7	1	_	9	_	_	<advance> (plate) (first)

It is important to note that not all actions are *legal* at every time t. For instance, in this example, it would be an *illegal* action for a verb to specify that a player advances from second to third since there is no player on second base at time $t-1$. This limitation forms the basis for one of our four tasks.

3.3 Definition of SPLAT Probing Tasks

The world-states and symbolic programs are not observable by the language model. Our objective is to measure the extent to which language models can construct and maintain world-states by learning latent versions of the symbolic program. To perform our evaluations, we present a language model with

a sequence of verbs $v_{0:t}$ and then present a question-answer pair about a world-state. We probe these capabilities with four types of questions:

- **Legal Verbs:** At an arbitrary time t we present the language model with the verb sequence $v_{0:t}$ as the input and evaluate the output distribution of the language model over the vocabulary for its ability to rank legal verbs higher than other verbs. This task measures the power of a language model to recognize *deterministic* constraints that may exist in the world-state (e.g., a pawn blocking the queen's path), making it distinct for the language modeling objective of assigning probability based upon a historical context.

- **Variable Assignment:** In this task the language model is presented with the verb sequence $v_{0:t}$, a question indicator and a label for variable e_n as input, while the probability distribution over possible assignments $a_k^{(n)}$ forms the output. With this probe it is possible reconstruct the entire state-space by iterating over all possible variables.

- **Who's Winning:** The language model is presented with the verb sequence $v_{0:t}$ and question indicator as input. The output of the language model is a probability distribution over special tokens representing *white*, *black* or *tied*. This task evaluates a language model's ability to synthesize information about assignments in a world-state at t. In chess, a weighted point value of pieces remaining on the board for each side is used to make this determination, while "[away]" and "[home]" scores are used for baseball. This information is only implicitly represented in the verb-encoding used by the language model.

- **Goal State:** In this task the language model is presented with the verb sequence $v_{0:t}$, a label for variable e_n and an assignment $a_k^{(n)}$ of a future world-state $S_{t+\Delta t}$. The language model output is a sequence of verbs $v_{t+1:t+\Delta t}$ which results in the assignment at time $(t + \Delta t)$. This task evaluates a language model's ability to make multi-hop decisions.

4 Datasets

4.1 The Chess Dataset

We derive our chess dataset from lichess.org, a free, open-source chess-playing site that attracts amateur and professional chess players. There are approximately 2.2 billion historical games published on the site. Each game consists of the series of moves made by each player, the game results, and selected metadata. Moves are expressed in standard algebraic notation (SAN), in which the destination coordinates encodes the move. When this is ambiguous, a character denoting the piece is prepended to the move. While compact, this notation requires knowledge of the current board configuration. To avoid this constraint, we elected to use universal chess interface notation (UCI), which specifies a move's starting and ending board coordinates. Additional characters are appended to denote castling and promotions. Having direct access to starting and ending coordinates is convenient for our evaluation tasks. Metadata includes information about the players, including Elo ratings, where a rating of 2000 or above is considered expert level.

We apply several filters to games from 2014 to arrive at a manageable yet representative chess dataset. We only consider games with more than ten and less than 150 moves where both players have Elo ratings of 2000 or above. We believe that this filtering methodology helps to ensure that we are training and evaluating on valid games. This results in a raw dataset of roughly 850,000 games, divided into pre-training and fine-tuning datasets of 400,000 games each, a test set of 25,000 games and a validation set of 25,000 games. (see Table 2 for summary statistics).

	Chess 400K	**Baseball 15K**
Number of Games	850K	32K
Avg. SPLAT Tokens per Game	73.5	704.2
Number of Pre-training Tokens	29.4M	10.6M
Number of Fine-tuning Questions	300K	73K
Vocabulary Size	4390	144
TLM Pre-training perplexity	6.8	2.0
LSTM Pre-training Perplexity	13.4	1.9

Table 2: **Dataset Summary Statistics.** The chess and baseball datasets used in this work are comparable in size to popular human-annotated question answering datasets. Question answer pairs for each dataset can be scaled by 10^3 without additional raw data.

4.2 The Baseball Dataset

Our baseball dataset is sourced from the Major League Baseball Gameday files. These files are publically available in XML format from 2008 through 2018 and contain detailed pitch-level plays for approximately 32,000 games. We parse the XML files into a sequence of nullary, unary and binary operators of our own construction. Each of these operators maps to deterministic changes in world-states (see Equation 1 and Figure 1).

We eliminate games with parsing errors or with sequence lengths greater than 1,000 to arrive at a dataset of 31,603 games. These are divided into pre-training and fine-tuning datasets of 15,000 games each, a test set of 800 games and a validation set of 800 games.

4.3 Comparison of Datasets

The coordinates of pieces on the board completely specify the world-state for chess. A sequence of *legal* moves, which are defined by the pattern in which piece can move and any constraints on destinations (*e.g.* a bishop can move an unlimited number of squares diagonally until it reaches the edge of the board, captures an opponent's piece or is block by a piece of the same color) express the gameplay of chess. Moves deterministically result in new piece locations and potentially a capture which removes an opponent's piece from the board. A game concludes when a player captures the opponent's king or the allotted game time expires.

The gameplay of baseball is distinct from chess, particularly in that the vocabulary of actions and world-state representations are more compact, while rules governing updates to world-states are more complex. Certain variables of the world-state will increment or reset conditioned upon the verb and other variables of the world-state. Vocabularies and world-state representations for chess and baseball are presented in Appendices A and B, respectively.[3]

5 Experiments and Results

5.1 Language Models

We probe the learning and representation capabilities of language models using three existing language model architectures. Since we formulate the evaluations as question-answer probes, our methodology is model-agnostic and can be applied to any language model capable of estimating the probability of a sequence of tokens up to the maximum sequence length of a single chess or baseball game.

We generate our main results using a transformer-based language model (TLM) based upon the original Transformer (Vaswani et al., 2017) using only the decoder side of the architecture as in (Radford and Narasimhan, 2018). A LSTM language model (Zaremba et al., 2014) is also evaluated. Although more sophisticated LSTM architectures exist (Merity et al., 2018; Yang et al., 2018), we elect to use a basic architecture since our goal is not to maximize performance but to identify which models exhibit competencies or deficiencies relative to a random baseline.

Lastly, we evaluate using GPT-2, which is architecturally analogous to the TLM. Our primary motivation in using GPT-2 is to evaluate the impact of pre-training on a large open-domain natural language corpus. Specifically, we seek to determine if a language model's ability to learn and represent chess can leverage the commonsense knowledge implicit in large training corpora.

5.2 Hyperparameters and Training

The language models are trained using established hyperparameters. The transformer-based language model uses a 512-dimensional embedding space with tied embeddings, eight attention heads and six attention layers, with a 0.1 dropout probability. The LSTM language model also uses a 512-dimensional embedding space with tied embedding, two LSTM layers, 30 time steps, a dropout rate of 0.5 and gradients are clipped at a norm of 2.0. The hyperparameters for these models closely follow the original architectures (Vaswani et al., 2017; Zaremba et al., 2014). We use the pre-trained GPT-2 small configuration.

Following established methodologies (Radford et al., 2019) we pre-train language models on a corpus of completed games and then fine-tune these models on partial games with question-answer pairs. These question-answer pairs are inserted at random times t. The transformer-based and LSTM language models are each pre-trained for 100 epochs and fine-tuned for 25 epochs. We also fine-tune from random initialization to evaluate the benefit of pre-training. The GPT-2 models are only

[3]Please see en.wikipedia.org/wiki/Rule_of_chess and https://en.wikipedia.org/wiki/Baseball for more detailed descriptions of chess and baseball, respectively.

	Chess Dataset					Baseball Dataset				
	Model Params	**Legal Verbs**	**Assign-ment**	**Who's Win.**	**Goal-state**	**Model Params**	**Legal Verbs**	**Assign-ment**	**Who's Win.**	**Goal-state**
TLM Pretrained	23.4	75.8				19.1	50.7			
TLM Finetuned	23.4	73.1	97.6	90.8	74.5	19.1	48.7	79.8	90.4	28.7
TLM Randomized	23.4	66.1	97.2	79.4	68.2	19.1	54.5	82.9	90.4	20.8
LSTM Finetuned	8.7	55.2	62.9	40.3	14.6	4.4	53.2	68.3	45.6	6.6
LSTM Randomized	8.7	47.9	63.8	63.7	15.9	4.4	52.8	74.3	49.4	10.9
GPT-2 Pretrained	115.0	5.0	2.0	23.3	n/a					
GPT-2 Finetuned	115.0	23.3	69.2	76.0	n/a					
Random Baseline		0.8	39.1	35.2	1.0		11.8	36.1	37.5	0.4

Table 3: **Probing Question Results.** Fine-tuned models outperform random baselines on the chess and baseball domains. Model parameters are reported in megabytes, legal verbs are reported as R-Precision percent, and all other metrics are in percent accuracy.

fine-tuned on the models pre-trained on a web-scale natural language corpus.

5.3 Experimental Setup

Each task is framed as a multiple-choice question-answer pair. In the case of the legal actions and goal-state tasks, the answer space is large and consists of all possible events.

To encode a question-answer pair about world-state S_t, we construct a sequence beginning with a [start] token, followed by all verbs $v_{0:t}$ up to location t, followed by a [sep] token, the question indicator and arguments, the answer indicator is terminated with an [end] token (see Appendices A and B for examples). During fine-tuning, we present these sequences to the model with "gold" answer tokens. For evaluation, we construct sequences with every possible answer and rank these according to the probability assigned to the each sequence by the language model.

For the legal action evaluation, R-Precision (Manning et al., 2005) is calculated over the top R verbs, where R is the number of possible legal verbs. For all other tasks, we evaluate accuracy defined by the number of correct answers divided by the number of questions. The world-state assignment (i.e., "Variable Assignment") and world-state synthesis task (i.e., "Who's Winning?") use the top-ranked assignment $a_k^{(n)}$ token as the answer. Actions for the goal-state task are generated using greedy decoding. The verb generated for $t-1$ is appended to the decoded sequence and is used to generate the verb at t, for up to a maximum of five predicted verbs.

Our evaluation framework is designed to be easily extensible. A number of "spare" indicators for questions, answers and values exist in the vocabulary for each dataset. It is simple to construct new question-answer pairs using the symbolic program to generate world-state S_t, analyzing this world-state to identify the "gold" answer and then encoding the entire sequence as described above.

5.4 Results

Our main results are shown in Table 3. The performance of fine-tuned models surpassed random baselines on both the chess and baseball datasets, providing strong support that language models can construct and maintain world-state representations and learn at least a portion of the functionality of the symbolic program for each domain. TLMs generally outperform LSTM language models, which may be related to the number of model parameters or differences in underlying architectures. Interestingly, TLMs consistently outperform GPT-2, despite almost identical architectures and fewer parameters. We construct random baselines by randomly sampling answers for each probing question on the validation set. Language models are trained on the 400K and 15K datasets for chess and baseball, respectively.

Language model performance on Legal Verbs is strong, with R-Precision for chess and baseball exceeding 50% for TLMs and LSTM. This is not surprising since this task is very similar to the language modeling objective used in pre-training. The difference in performance between the chess and baseball datasets may result from the pre-training dataset for chess being approximately $3x$ larger. Also, the gameplay of baseball contains several rare and hard to predict events, such as stolen bases and player substitutions.

Performance on the Variable Assignment task is also very strong for language models in each domain. This task is the most direct measure of a language model's ability to construct and maintain world-states. Prior works on chess (Toshniwal et al., 2021) perform a similar task but evaluate only on the piece location resulting from the most recent move. We select arbitrary world-state variables e_n regardless of how recently the assignment $a_k^{(n)}$ was made.

The transformer-based language models (including GPT-2) performed better on the Who's Winning task than the LSTMs. This suggests that an explicit attention mechanism may be helpful when making implicit calculations and comparisons over world-states.

Our most unexpected result is the strong TLM performance on the Goal State task for chess. We would expect it to be non-trivial to decode a sequence of up to five moves that achieves a given goal. Conversely, task performance on baseball is worse than we expected, particularly given that there are a wider variety of non-unique paths leading to the same world-state. For example, many combinations of balls, strikes, fouls, etc. could result in a batter being out. Additional analysis is needed to explain these performance differences and is an item of future research.

We evaluated on GPT-2 to determine the extent to which the model already "knows how to play chess" based upon pre-training on a web-scale natural language corpus. Results using UCI chess notation are not materially different from baselines. We did not evaluate GPT-2 on the Goal State task for chess since it would require approximately 10^{15} inference passes. We did not evaluate GPT-2 on the baseball dataset since our SPLAT encoding is unique to this dataset.

Lastly, performance on Legal Verbs for the fine-tuned TLM models is lower than for the pre-trained TLM models. This indicates that fine-tuning can sometimes cause the pre-trained model to forget, which is consistent with other findings (Ott, 2021).

6 Discussion and Analysis

To better understand two important aspects of our results, we conduct additional analyses of (i) a language models' ability to learn the "rules" of each game, and (ii) the extent to which memorization drives performance on probing tasks.

6.1 Legal Action Constraints

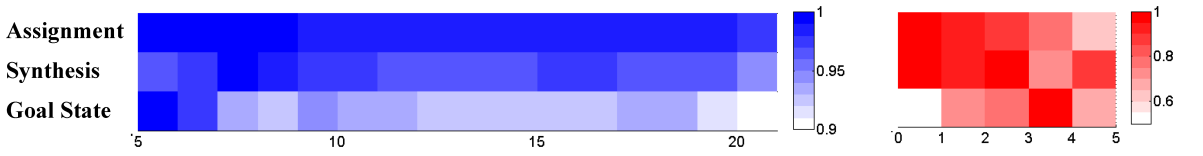
In our experiments above we evaluate the ability of language models to learn the "rules" of a world-state by measuring the R-Precision of legal actions at an arbitrary time t conditioned on the previous world-state S_{t-1}. This measures the ability of the language model to rank all legal actions higher than all other actions collectively.

	TLM Pre	TLM Fine	LSTM Fine	GPT-2 Fine
Legal (r-prec)	75.8	73.1	55.2	23.3
In-Pattern (mass)	97.8	93.7	57.5	30.0
Legal (mass)	95.8	91.9	46.2	25.4
Blocked (mass)	2.0	1.7	11.3	4.6

Table 4: **Error Analysis of Legal Actions on the Chess Dataset.** Language models assign high probability mass to Legal Moves. At the same time, the R-Precision of Legal Moves is lower, indicating that language models may be focusing on the "best" moves rather than all possible moves. The low probability mass assigned to Blocked Moves, calculated as the difference between probability masses assigned to In-Pattern Moves and Legal Moves, indicates that the language model may be aware of rules constraining moves based upon the current world-state.

This evaluation may form an incomplete picture. The language model may allocate a large portion of the probability mass to a few verbs representing the most likely moves. This allocation is consistent with the language model training objective, where the model learns from only "positive" examples presented during training, and empirical next-move distribution is typically highly peaked on a few likely choices. The training data does not include many unlikely moves, which may make it difficult for the models to learn to distinguish between unlikely and truly impossible moves that would violate the rules, lowering legal-move R-precision. This appears to be the case with transformer-based language models where the probability mass assigned to legal actions is above 90%, while R-Precision is lower (see Table 4).

The probability mass assigned to blocked actions is a complementary metric to R-Precision. We define Blocked actions as actions that would otherwise be legal except for some constraint in the world-state. One example is a pawn blocking the trajectory of its own queen. We measure the probability mass assigned to blocked actions by subtracting the probability mass assigned to Legal Moves for a given world-state S_t from the In-



(a) Chess Dataset **(b) Baseball Dataset**

Figure 2: **Memorization Analysis.** Average accuracy (darker is more accurate) on probing tasks for chess and baseball does not materially decline as the distance to the closest world-state in the fine-tuning dataset increases, indicating that memorization is not a strong factor in model performance. We calculate chess dataset accuracy as the rolling average of the previous five distance metrics to reduce noise. On the baseball dataset, we normalize by the maximum performance on each task.

Pattern probability mass assigned to actions in a world-state without blocked moves (i.e., as if the moving piece were alone on an otherwise empty chessboard). We see that the transformer-based language model assigns a small probability to blocked actions.

6.2 Memorization

Large transformer-based language models are often introduced simultaneously with new web-scale corpora. This makes it difficult to determine if performance gains are a result of an improved model architecture or simply having additional content to memorize. It is challenging to test this hypothesis given the unstructured and open-domain nature of these corpora. The overlap of 8-grams among the training and test sets is examined in one recent work (Radford et al., 2019). We seek to build upon this approach by analyzing memorization more directly, which is possible in our restricted SPLAT domains.

To evaluate the impact of memorization on probing questions, we construct a distance metric between the world-state S_t at the time of the probing question and the closest world-state observed in the fine-tuning dataset. We examine how this distance varies with question-answering accuracy, across the Variable Assignment, Who's Winning, and Goal State tasks. If results are driven primarily by memorization, we expect to see significant attenuation in performance as distance increases. This is not generally the case (Figure 2). We see some attenuation for Goal State, but for the other two tasks performance stays high even for the world-states most distant from those in the fine-tuning dataset. Similar results are obtained for baseball at the 50% accuracy threshold, although the pattern of results is less regular than for baseball.

7 Conclusion

We introduce the SPLAT approach and novel chess and baseball datasets, and use them to probe language models' learning and representation capabilities. We perform these probes by pre-training language models on SPLAT encodings of gameplay and then fine-tuning on question-answers pairs constructed to evaluate language model capabilities. Probes are designed to evaluate a language model's ability to learn the "rules" of each game, construct and maintain world-states, synthesize across states and generate multi-step paths to each a given goal. We show that language models of different architectures possess the learning and representation capabilities to perform well on these tasks.

These datasets are different from existing datasets in that (i) question-answer pairs can be generated at scale while maintaining the natural distribution properties of the gameplay data, (ii) accompanying symbolic programs can map verbs to deterministic changes in world-state, and (iii) complete knowledge is available throughout the question-answering process. We demonstrate a probing methodology that is extensible to different language models and additional probing questions. Lastly, we note that SPLATs can be constructed for other domains such as WOZ dialogues (Budzianowski et al., 2018) or quantitative behavior finance (Ramiah et al., 2015).

Acknowledgments

This work was supported in part by a Microsoft Research Faculty Fellowship and NSF grant IIS-2006851. We thank the anonymous reviewers for their insightful comments.

References

Paweł Budzianowski, Tsung-Hsien Wen, Bo-Hsiang Tseng, I. Casanueva, Stefan Ultes, Osman Ramadan, and Milica Gasic. 2018. Multiwoz - a large-scale multi-domain wizard-of-oz dataset for task-oriented dialogue modelling. *ArXiv*, abs/1810.00278.

Danqi Chen, Adam Fisch, J. Weston, and Antoine Bordes. 2017. Reading wikipedia to answer open-domain questions. In *ACL*.

Grzegorz Chrupała and A. Alishahi. 2019. Correlating neural and symbolic representations of language. In *ACL*.

O. David, N. Netanyahu, and L. Wolf. 2016. Deepchess: End-to-end deep neural network for automatic learning in chess. *ArXiv*, abs/1711.09667.

Suchin Gururangan, Swabha Swayamdipta, Omer Levy, Roy Schwartz, Samuel R. Bowman, and Noah A. Smith. 2018. Annotation artifacts in natural language inference data. In *NAACL*.

Dieuwke Hupkes and Willem H. Zuidema. 2018. Visualisation and 'diagnostic classifiers' reveal how recurrent and recursive neural networks process hierarchical structure. *ArXiv*, abs/1711.10203.

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

Chaitanya Malaviya, Chandra Bhagavatula, Antoine Bosselut, and Yejin Choi. 2020. Commonsense knowledge base completion with structural and semantic context. In *AAAI*.

Christopher D. Manning, P. Raghavan, and Hinrich Schütze. 2005. Introduction to information retrieval.

Bryan McCann, N. Keskar, Caiming Xiong, and R. Socher. 2018. The natural language decathlon: Multitask learning as question answering. *ArXiv*, abs/1806.08730.

Stephen Merity, N. Keskar, and R. Socher. 2018. Regularizing and optimizing lstm language models. *ArXiv*, abs/1708.02182.

Myle Ott. 2021. Analyzing the forgetting problem in pretrain-finetuning of open-domain dialogue response models. In *EACL*.

Alec Radford and Karthik Narasimhan. 2018. Improving language understanding by generative pre-training.

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

Colin Raffel, Noam M. Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, W. Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *ArXiv*, abs/1910.10683.

Pranav Rajpurkar, Robin Jia, and Percy Liang. 2018. Know what you don't know: Unanswerable questions for squad. In *ACL*.

Vikash Ramiah, Xiaomin Xu, and I. Moosa. 2015. Neoclassical finance, behavioral finance and noise traders: A review and assessment of the literature. *International Review of Financial Analysis*, 41:89–100.

Siva Reddy, Danqi Chen, and Christopher D. Manning. 2019. Coqa: A conversational question answering challenge. *Transactions of the Association for Computational Linguistics*, 7:249–266.

Anna Rogers, Olga Kovaleva, and Anna Rumshisky. 2020. A primer in bertology: What we know about how bert works. *Transactions of the Association for Computational Linguistics*, 8:842–866.

Julian Schrittwieser, Ioannis Antonoglou, Thomas Hubert, K. Simonyan, L. Sifre, Simon Schmitt, A. Guez, Edward Lockhart, D. Hassabis, T. Graepel, T. Lillicrap, and D. Silver. 2020. Mastering atari, go, chess and shogi by planning with a learned model. *Nature*, 588 7839:604–609.

Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. 2019. Commonsenseqa: A question answering challenge targeting commonsense knowledge. In *NAACL-HLT*.

Ian Tenney, Dipanjan Das, and Ellie Pavlick. 2019. Bert rediscovers the classical nlp pipeline. In *ACL*.

Shubham Toshniwal, Sam Wiseman, Karen Livescu, and Kevin Gimpel. 2021. Learning chess blindfolded: Evaluating language models on state tracking. *ArXiv*, abs/2102.13249.

Ashish Vaswani, Noam M. Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *ArXiv*, abs/1706.03762.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. 2018. Glue: A multi-task benchmark and analysis platform for natural language understanding. In *BlackboxNLP@EMNLP*.

J. Weston, Antoine Bordes, S. Chopra, and Tomas Mikolov. 2016. Towards ai-complete question answering: A set of prerequisite toy tasks. *arXiv: Artificial Intelligence*.

Zhilin Yang, Zihang Dai, R. Salakhutdinov, and William W. Cohen. 2018. Breaking the softmax bottleneck: A high-rank rnn language model. *ArXiv*, abs/1711.03953.

Wojciech Zaremba, Ilya Sutskever, and Oriol Vinyals. 2014. Recurrent neural network regularization. *ArXiv*, abs/1409.2329.

Rowan Zellers, Yonatan Bisk, Roy Schwartz, and Yejin Choi. 2018. Swag: A large-scale adversarial dataset for grounded commonsense inference. In *EMNLP*.

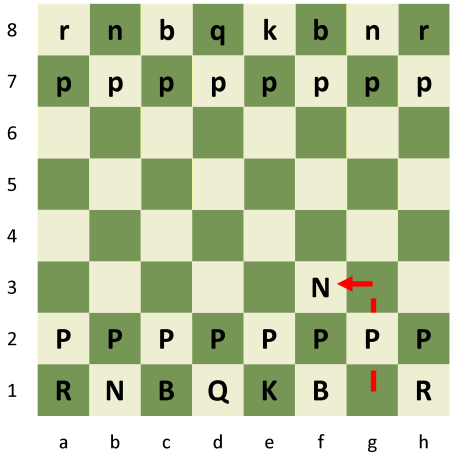
A. Overview of Chess Dataset

	Vocabulary	Count
Moves	a2a3, a2a4,…h7h6	4,272
Pieces	R, N, B, Q, K, P, r, n, b, q, k, p, .	13
Squares	a1, a2, … h8	64
Special Tokens	[start], [sep], [end], 0-1c, 1-0c, … 1/2-1/2n	9
Questions	Q1, Q2, … Q16	16
Answers	A1, A2, … A16	16
		4,390

(a) Vocabulary of Chess Dataset

Notation Example: SAN: Nf3 UCI: g1f3

(b) Notation Variations to Move Knight



(c) Chess World-State and Coordinates

Natural Language Question: Which piece is in square *d7* after the 21st move of the game?

Fine-tuning Sequence: `[start] d2d4 g8f6 ... g5h4 b8d7 h4g3 a7a6 a1c1 [sep] Q3 d7 n [end]`

Inference Sequence: `[start] d2d4 g8f6 ... g5h4 b8d7 h4g3 a7a6 a1c1 [sep] Q3 d7 <?>`

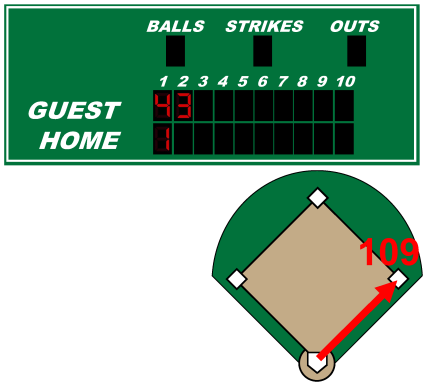
(d) Question Encoding Example

Figure 3: **Chess Representations and Question Encoding Example.** The gameplay of chess is encoded with 4,390 specialized tokens and forms the vocabulary for this dataset (see Panel a). Verb-like tokens in UCI notation encode moves by specifying the "from-square" and "destination square" coordinates using two characters each and account for most of the vocabulary. Canonical alphabetic letters designate pieces, with capital and lower-case letters signifying white and black pieces, respectively, and periods encoding *Empty* squares. Squares are mapped to the 2 x 2 coordinate grid of the board, as shown in Panel (c). Special tokens designate starting, ending and separation boundaries for question-answer pairs and include canonical tokens identifying the winning player or tied and whether the game-ending was "normal" or a "clock forfeiter". Single indicator tokens designate questions and answers, where such encoding maintains the closed domain nature of the dataset. During fine-tuning, a partial game sequence is presented to the language model, followed by a separator token, a question indicator (including arguments) and an answer token or indicator (see Panel d). During inference, the answer token or indicator is replaced with all possible answers for the question (illustrated and $<?>$ above). The language model assigns a probability to each sequence and selects the most probable sequence as the answer.

B. Overview of Baseball Dataset

	Vocabulary	Count
Nullary Verbs	<strike>, <ball>, <foul>, <out>, <atbat>, <hit>, <sog>, <eog>	8
Unary Verby	<newbatter> (player_id) <caught_stealing> (location)	2
Binary Verbs	<sub> (player_id) (player_id) <advance> (from_base) (to_base)	2
Elements	[inning], [half], [balls], [strikes], [outs], [away], [home], [plate], [first], [second], [third]	11
Special Tokens	[start], [sep], [end], '___'	4
Locations	(plate), (first), (second), (third), (score)	5
Player IDs	(id_100, id_101,… id_147)	48
Questions	Q1, Q2, … Q16	16
Answers	A1, A2, … A16	16
Values	0, 1, … 31	32
		144

(a) Vocabulary of Baseball Dataset



(b) Traditional Baseball Representation

Balls	Strikes	Outs	Inning	Half	Away	Home	Plate	First	Second	Third	
0	0	0	2	B	7	1	9	_	_	_	<hit>
0	0	0	2	B	7	1	_	9	_	_	<advance> (plate) (first)

(c) World-State Elements and Assignments

Natural Language Question: Who's on first?

Fine-tuning Sequence: [start] <sog> ... <newbatter> (id_109) <ball> <ball> <strike> <hit> <advance> plate)(first) [sep] Q1 (first) (id_109)[end]

Inference Sequence: [start] <sog> ... <newbatter> (id_109) <ball> <ball> <strike> <hit> <advance> (plate)(first) [sep] Q1 (first) (?)

(d) Question Encoding Example

Figure 4: **Baseball Representations and Question Encoding Example.** The gameplay of baseball is encoded with 144 tokens and forms the vocabulary for the for the dataset (see Panel a). Events that occur during gameplay are encoded as a set of nullary, unary and binary verbs (including associate arguments). Variable represent labels for each variable in the world-state. Possible assignments for each variable of a world-state include Locations, Player IDs and Values. Special tokens designate starting, ending and separation boundaries for question-answer pairs, and also includes a '___' token for an empty base. The world-state of baseball is commonly presented as a snapshot of the scoreboard and field (see Panel b). We represent these world-states as a set of eleven variables (see Panel c). These gameplay and world-state representations are formulated specifically for this dataset. Single indicator tokens designate questions and answers, where such encoding maintains the closed domain nature of the dataset. During fine-tuning, a partial game sequence is presented to the language model, followed by a separator token, a question indicator (including arguments) and an answer token or indicator (see Panel c). During inference, the answer token or indicator is replaced with all possible answers for the question (illustrated and $<?>$ above). The language model assigned a probability to each sequences and selects the most probable sequence as the answer.