languages

MDPI

Article

BabyDS: Visually Grounded Grammar Induction with Online Curriculum Learning

Arash Ashrafzadeh [1,*], Julian Hough [2] and Arash Eshghi [1]

[1] Department of Computer Science, Heriot-Watt University, Edinburgh EH14 4AS, UK; a.eshghi@hw.ac.uk
[2] Department of Computer Science, School of Mathematics and Computer Science, Swansea University, Swansea SA1 8EN, UK; julian.hough@swansea.ac.uk
* Correspondence: aa2070@hw.ac.uk

Abstract

Recent research in grounded language learning has seen remarkable success due to advances in large vision and language models (VLMs). However, these models (i) are extremely costly to train and update; (ii) struggle with generalisation; and (iii) do not support *continual learning*. In this paper, we introduce BABY-DS integrating the Dynamic Syntax (DS) framework with automated planning within the multimodal BabyAI platform as a testbed. We provide methods whereby DS lexicons are induced continually from teacher demonstrations within BabyAI. We study (i–iii) by experimenting with the compositional complexity of natural language instructions in the data to compare data efficiency, generalisation, and continual learning properties of BABY-DS with a simple neural model. The results show that the BABY-DS model: (i) needs much less data than the neural model to reach threshold performance; (ii) generalises much faster to more complex instructions; and (iii) is a more effective continual learner. We argue that it is the attendant linguistic bias within DS and the rich inferential power of TTR that enables (i–iii), highlighting the importance of further research on hybrid grammar–neural approaches. Finally, we discuss several important limitations of BABY-DS and sketch a path forward for further DS research.

Keywords: grammar induction; neural semantic parsing; computational semantics; grounded language learning

1. Introduction

Grounded language learning addresses a fundamental challenge in artificial intelligence: moving beyond abstract language understanding, which is typically learned from text-only data, to enable models to truly comprehend and use language by connecting symbols to real-world perceptual experiences. Unlike models trained solely on text, which—despite capturing rich distributional patterns (Bommasani et al., 2021)—lack grounding in perceptual experience, grounded language learning aims to address the *symbol grounding problem* (Harnad, 1990), where meaningless symbol tokens are given meaning through direct experience with objects and events in a given physical context. This is crucial because language symbols acquire meaning when actively used to achieve goals and coordinate with others in the world, much like humans learn through "language games" in the Wittgensteinian sense (Wittgenstein, 1953), i.e., situated communicative activities in which linguistic meaning is constituted through use (see Suglia et al. (2024) for a comprehensive review).

Academic Editor: Julien Longhi

Received: 29 October 2025
Revised: 13 April 2026
Accepted: 14 April 2026
Published: 12 May 2026

The problem of language learning in multi-modal interactive environments has received significant attention, especially with the rise of Vision–Language Models (VLMs) and Reinforcement Learning (RL). Developing agents that can understand and generate natural language in interactive, situated, and embodied environments is vital for creating intelligent systems that can resolve ambiguities, understand the semantics of situations and actions, and ultimately engage in cooperative communication with humans, which is essential for user-facing, situated AI systems including in robotics. Researchers have tackled this idea within various frameworks, from simple 2-D environments to complex 3-D ones and extending to embodied systems acting in real-world controlled environments (Hermann et al., 2017; Misra et al., 2017 among many others), all attesting to the significance of this task.

Within grounded language learning test-beds, BabyAI (Chevalier-Boisvert et al., 2018), an instruction-following environment initially created to study the data efficiency of models in a controllable, situated setup, has remained a challenge for state-of-the-art (SOTA) neural models. Previous research (see Section 1.1) has employed mainly VLMs and RL techniques (see Carta et al., 2023; Chevalier-Boisvert et al., 2018; Fu et al., 2024; Mirchandani et al., 2021, among others), achieving strong results. However, there is yet no systematic study of these models that relates their data efficiency and generalisation properties to their architecture and overall methodology (e.g., grammar-based vs. end-to-end neural models or symbolic vs. vector-based semantics).

Even within the very simple structure of the BabyAI environment and the attendant natural language (NL) instructions, BabyAI models achieve good results only when trained on unrealistically large amounts of BabyAI data, and thus exhibit very poor data efficiency properties (Chevalier-Boisvert et al., 2018; Hui et al., 2020). This remains true even of VLM models already pretrained on large amounts of diverse data from all over the internet. Therefore, such models do not seem to satisfy the initial purpose BabyAI was created for: that of building data-efficient models with human-like generalisation capabilities to enable human-in-the-loop training.

Crucially, our primary focus here is a degree of data efficiency and compositional generalisation that enables human-in-the-loop training. It is well-documented that pre-trained open-source VLMs, that can potentially be fine-tuned, struggle significantly with data efficiency and compositional generalisation (Baroni, 2020; W. Chen et al., 2026; Pantazopoulos et al., 2022), and seemingly find it impossible to achieve the order of data efficiency and generalisation we target, providing strong justification for why we do not include a VLM or Transformer baseline in our experiments.

In this paper, with the above in mind, we develop a symbolic grammar-based model for the BabyAI environment based on the Dynamic Syntax and Type Theory with Records (DS-TTR) grammar framework (Cann et al., 2005; Eshghi et al., 2012; Kempson et al., 2001; Purver et al., 2011), where the symbolic semantic output of parsing is grounded in aspects of the BabyAI environment, following a types-as-classifiers approach (Hough et al., 2020; Larsson, 2013; Yu et al., 2016). We go on to extend and adapt the grammar induction method of Eshghi et al. (2013a) to enable *continual learning*, and apply it to the problem of grounded language learning within BabyAI.

We then use the controllable BabyAI environment and our own experimentally controlled training and testing scenarios to study the following research questions: RQ1—data efficiency: How fast does our grammar-based model learn to solve BabyAI tasks and how does its data efficiency relate to their complexity? How does the grammar-based model compare to a simple neural model trained and tested in identical conditions? RQ2—generalisation and forgetting: In a fair experimental setup, how fast does our grammar-based model generalise to unseen, but partially overlapping data and how does

this compare to the neural model? How good are the models at remembering previously learned patterns when trained on new, unseen but partially overlapping data in a continual learning scenario?

The results of our experiments show that the DS-TTR models exhibit much faster learning and generalisation properties than the neural models. We argue that, as these models stand, only the former is suitable in a realistic human-in-loop training scenario.

Given the inherent incrementality in language processing within DS-TTR itself and the continual learning capability of our grammar induction method, we have arguably taken an important step towards building models of real-time, grounded language learning, suitable for deployment in realistic human-in-the-loop teaching scenarios.

Finally, we discuss the various limitations of this work, and provide suggestions for further expansion of this line of research.

1.1. Related Work

In this section, we first very briefly introduce the BabyAI environment, then review the various approaches for solving its missions. We then move on to discuss some of the relevant work on grammar-based grounded lexical learning.

1.1.1. The BabyAI Environment

First introduced by Chevalier-Boisvert et al. (2018), BabyAI—described in greater detail below in Section 2.1—is a simple grid world environment with various objects in different locations where an agent has to follow instructions in NL to solve various 'missions', by taking a sequence of limited, discrete, atomic actions made available to it by the environment (e.g., navigation actions)—see Figure 1. With missions being systematically organised in terms of levels of difficulty, the environment is designed with the express purpose of "studying the sample efficiency of grounded language learning" models and to "support investigations towards including humans in the loop for grounded language learning" (Chevalier-Boisvert et al., 2018). As such, the platform is highly suited to investigating the research questions set out above in the Introduction.

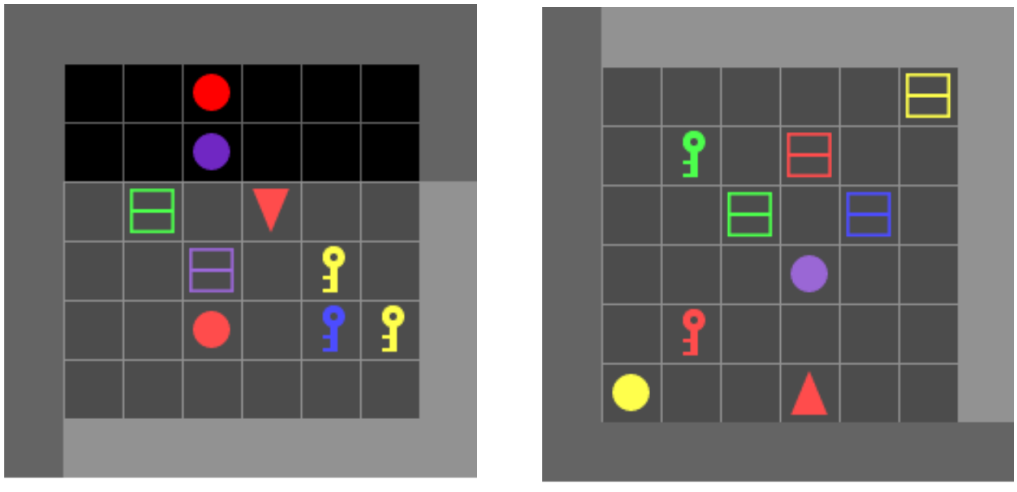


Figure 1. Two examples of initial states from the BabyAI environment together with example NL instructions. The red arrow represents the location and direction of the agent in the grid.

1.1.2. Neural Models for BabyAI

Table 1 organises the papers that have used the BabyAI environment as a testbed for the models they provide. The publications are organized according to three main

dimensions: type of model, performance measures, and the evaluation scenarios used in each paper. We now move on to discussing some of the work in greater detail.

Table 1. Summary of previous models on BabyAI, their properties and limitations. Abbreviations: TSR: Task Success Rate, SA: Semantic Accuracy, CL: Curriculum Learning, IL: Imitation Learning, CG: Compositional Generalisation, SE: Sample Efficiency, OL: Online Learning, RL: Reinforcement Learning, and LLM: Large Language Model.

Model	Approach	Performance Measures	Evaluation Scenarios
BabyAI 1.0 (Chevalier-Boisvert et al., 2018)	RL	TSR	CL + IL
BabyAI 1.1 (Hui et al., 2020)	RL	TSR	CL + IL
ELLA (Mirchandani et al., 2021)	RL	TSR	CG, SE, OL
GLAM (Carta et al., 2023)	LLM + RL	TSR	CG, SE, OL
LAST (Fu et al., 2024)	LLM	TSR	CG, SE, OL
BABY-DS (ours)	Parsing + Planning	SA + TRS	SE, CL, OL

Chevalier-Boisvert et al. (2018) developed the original LSTM-based baselines for BabyAI, revealing that the deep learning methods of the time require vast amounts of data to solve tasks end-to-end, though curriculum learning and interactive teaching offer measurable improvements. Building on this, Hui et al. (2020) introduced architectural modifications, such as removing max-pooling and adding residual connections, that improved RL sample efficiency; however, due to rapid advances in the field, these early baselines have not remained state-of-the-art.

Subsequent work focused primarily on improving sample efficiency through reward shaping. Mirchandani et al. (2021) proposed ELLA, a two-classifier reward-shaping approach that, while effective, requires an upfront data budget and struggles with highly compositional instruction spaces. Addressing this limitation, Carta et al. (2022) introduced EAGER, which automates reward shaping via a Question Generation and Answering system, eliminating the need for expert-designed auxiliary objectives but still requiring a pre-trained QA module bootstrapped from successful trajectories.

In a different direction, Spilsbury and Ilin (2022) tackled compositional generalisation using Sparse Factored Attention with a Value Propagation Network planner, achieving success with only 50 samples per goal on BabyAI's GoToLocal missions; however, this approach relies on strong assumptions such as fully observable, disentangled inputs, limiting its applicability to more complex goals. Shifting to LLM-based approaches, Carta et al. (2023) introduced GLAM, which finetunes a FLAN-T5 LLM as an agent policy using online PPO in a textual version of BabyAI, demonstrating improved sample efficiency and generalisation to new objects. Most recently, McCallum et al. (2025) proposed FOSSIL, a Transformer-based imitation learning approach that contextualises actions with constructive language feedback, achieving success rates over four times higher than standard baselines, though at the cost of substantially larger model and training data sizes compared to our approach (see Section 3.2).

Despite these advances, none of these neural approaches have achieved a level of data efficiency sufficient for human-in-the-loop collaborative learning. Moreover, previous research has cast serious doubt on the compositional generalisation and data efficiency capabilities of neural models, including large VLMs and Transformer architectures (Baroni, 2020; W. Chen et al., 2026; Pantazopoulos et al., 2022). Therefore, the order of data efficiency and generalisation we are seeking seems to be impossible for VLMs to yield. This constitutes our solid motivation for comparing our approach against a simple, LSTM baseline rather than a Transformer-based VLM, allowing us to evaluate the core differences between a grammar-based and an end-to-end neural approach under identical, data-restricted training conditions. To date, no existing study evaluates all neural models under identical

training data and testing scenarios that go beyond task success rate to probe generalisation, catastrophic forgetting, and compositional generalisation while controlling for the amount of pre-training; conducting such an evaluation is therefore a contribution of our work. While large neural models have seen great success on many tasks, their size, data requirements, poor generalisation, catastrophic forgetting, and non-incrementality remain far from the characteristics of human language acquisition (Eshghi et al., 2022), motivating our investigation of grammar-based approaches.

1.1.3. Grammar-Based Grounded Lexical Learning

Among existing grammar-based approaches to language learning, Combinatory Categorial Grammar (CCG, Steedman, 1996, 2000) and Dynamic Syntax (DS, Cann et al., 2005; Kempson et al., 2001) have received the most attention. CCG in particular has been widely applied to wide-coverage parsing and grounded language learning.

In early work, Matuszek et al. (2012) jointly induced attribute classifiers and compositional semantic parsers using an EM-like algorithm, though the approach relies on a small supervised bootstrap dataset and suffers from limited grammar coverage. Similarly targeting grounded language acquisition, D. Chen (2012) proposed the SGOLL algorithm for efficiently aligning natural language with meaning representations in a navigation task, achieving strong performance but with scalability limited by the downstream KRISP parser. Extending this line of research to weak supervision, Artzi and Zettlemoyer (2013) developed a grounded CCG parser that learns from observing final execution states, completing 60% more instruction sets than prior work, though with exponential worst-case complexity. In a related vein, Ross et al. (2018) learned grounded semantic parsers from captioned videos, jointly resolving visual and linguistic ambiguities but limited by strict evaluation metrics and unreliable computer vision. Most recently, Mao et al. (2021) introduced G2L2, a neuro-symbolic CCG-based approach that learns grounded lexicon entries via a novel joint parsing and expected execution algorithm, demonstrating strong data efficiency and compositional generalisation on visual reasoning and navigation tasks.

Although the systems just described have achieved impressive results in terms of coverage, the grammar is not inherently incremental and therefore not suited for word-by-word processing of natural language. On the other hand, Dynamic Syntax (DS) is an inherently incremental framework but the computational work in DS is more limited. Among other fundamental work, Eshghi et al. (2013a, 2013b) introduced the first DS-TTR probabilistic grammar induction on simple child-directed utterance using the Expectation Maximisation algorithm for wide-coverage parsing. Although their methodology is going to shape the core of our induction algorithm, their model lacks grounding and has not been challenged with linguistically complex sentences. Furthermore, Shalyminov et al. (2017) and Eshghi et al. (2017) report strong generalisation and data efficiency in task-oriented dialogue, outperforming the then-SOTA MemN2N model of Bordes et al. (2017) thanks to the inductive bias of their grammar-based approach. However, the learned lexicon in their work is not grounded in a multi-modal environment. Finally, Yu et al. (2017) present a multimodal dialogue system that learns visually grounded word meanings through interactive conversation with a simulated human tutor, using reinforcement learning to optimise dialogue policies that balance classification accuracy with tutoring costs. Although their modular model achieves great data efficiency in grounded lexical learning, the effect of online curriculum learning on performance and compositional generalisation has not been explored in this or any of the previous work in DS.

1.2. Research Questions

To address the several gaps in the literature explained above, we use a controllable artificial agent in the BabyAI environment and our own experimentally controlled training and testing scenarios to address the following research questions:

RQ1—data efficiency: How fast can a DS grammar-based model learn to solve BabyAI tasks and how does its data efficiency relate to their complexity? How does the grammar-based model compare to a simple neural model trained and tested in identical conditions?
RQ2—generalisation and forgetting: In a fair experimental setup, how fast does our grammar-based model generalise to unseen, but partially overlapping data and how does this compare to the neural model? How good are the models at remembering previously learned patterns when trained on new, unseen but partially overlapping data in a continual learning scenario?

2. Materials and Methods

2.1. The BabyAI Platform

First introduced by Chevalier-Boisvert et al. (2018), BabyAI is a simple grid world environment where an instruction-following agent has to succeed in missions by taking actions (simply limited to `forward, left, right, pick-up, drop & toggle`) towards a goal (potentially made up of "subgoals"), and was initially created to explore the sample efficiency of grounded language learning models. Missions come in 19 levels of difficulty based on the simplicity of the instruction given its linguistic features and also the environmental complexity (size of the room(s) and the number, colour and type of the objects in it). The instructions are generated based on a BNF grammar, where the structure complexity of instructions can vary (e.g., from "go to the red box" to "pick up a key and put the green box next to the yellow ball, before you open the purple door and go to the box behind you"—see Figure 1). For more information on the BNF grammar generating the mission instructions, implementation details, etc., please refer to Chevalier-Boisvert et al. (2018).

BabyAI has been adapted by many researchers for different purposes, and below we review some of this work.

2.2. Dynamic Syntax

Dynamic Syntax (DS, Cann et al., 2005; Kempson et al., 2016, 2001) is an action-based grammar framework that directly captures the time-linear, incremental nature of the dual processes of linguistic comprehension and production, on a word-by-word or token-by-token basis. It models the linear construction of *semantic* representations (i.e., *interpretations*) as progressively more linguistic input is parsed or generated. DS is idiosyncratic in that it does not assume, or recognise, an independent level of syntactic representation over words: syntax on this view is sets of constraints on the incremental processing of semantic information in potentially multiple modalities (e.g., language and vision).

The output of parsing any given string of words, or non-verbal tokens, is thus a *semantic* tree representing its predicate-argument structure. DS trees are always binary branching, with argument nodes conventionally on the right and functor nodes to the left. Binary branching is standard in semantic grammar frameworks—it is shared, for example, by Combinatory Categorial Grammar (Steedman, 2000)—as it enables the progressive composition of individual word meanings into the meaning of overarching phrases or sentences. Unlike CCG, however, DS does not posit an independent level of syntactic representation; binary branching here directly reflects semantic composition. Tree nodes correspond to terms in the lambda calculus, decorated with labels expressing their semantic type (e.g., $Ty(e)$) and logical formulae—here as record types of Type Theory with Records (TTR, see Section 2.3 below); and beta-reduction determines the type and formula at a mother node from those

at its daughters (see Figure 2 for an example lexical action, and Figure 3 for the complete incremental parse of 'John arrives'). These trees can be *partial*, containing unsatisfied *requirements* potentially for any element (e.g., $?Ty(e)$, a requirement for future development to $Ty(e)$), and contain a *pointer*, $\Diamond$, labelling the node currently under development.

Grammaticality is defined as parsability in a context: the successful incremental word-by-word construction of a tree with no outstanding requirements (a *complete* tree) using all information given by the words in a string. We can also distinguish *potential grammaticality* (a successful sequence of steps up to a given point, although the tree is not complete and may have outstanding requirements) from *ungrammaticality* (no possible sequence of steps up to a given point) (Cann et al., 2007).

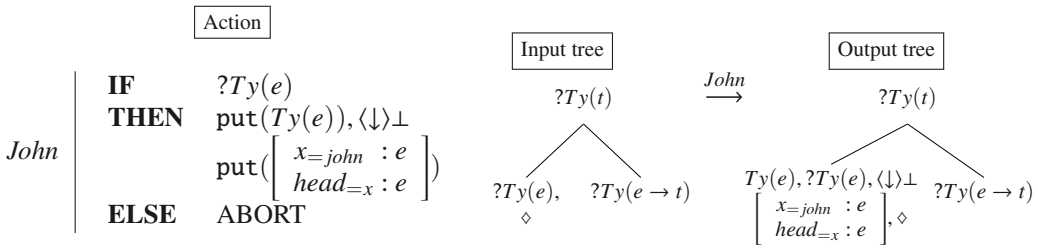


Figure 2. Lexical action for the word 'John'.

Figure 3 shows "John arrives", parsed incrementally, starting with an empty tree, with only the root node's daughters created, and ending with a complete tree. The intermediate steps show the effects of COMPLETION, which moves the pointer up and out of a complete node; and of ANTICIPATION, which moves the pointer down from the root to its functor daughter.

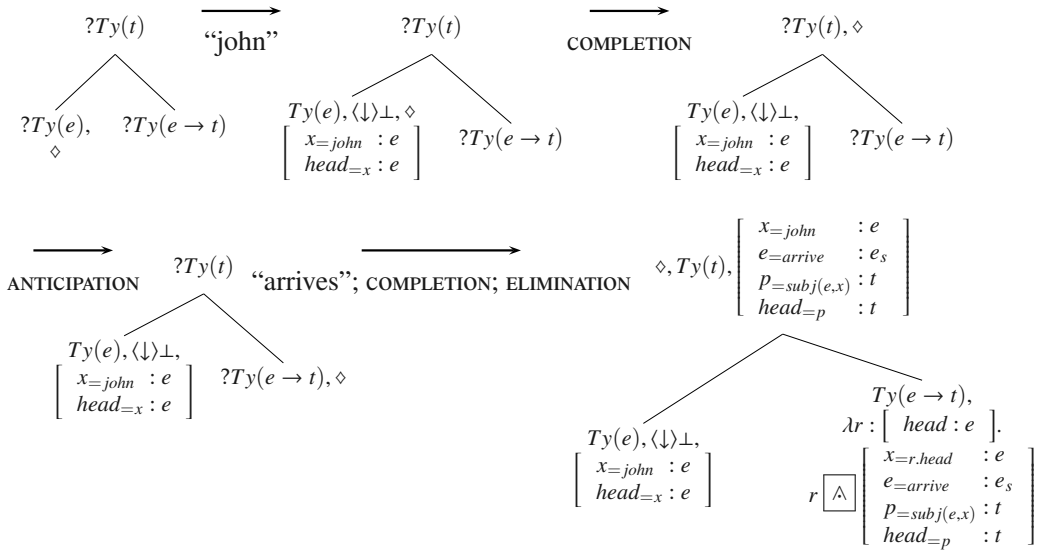


Figure 3. Incremental parsing in DS-TTR: *"John arrives"*.

Actions in DS

The parsing process, which is what constitutes 'syntax' in DS, is defined in terms of conditional *actions*: procedural specifications for monotonic semantic tree update. *Computational actions* are general structure-building principles; and *lexical actions* are language-specific actions corresponding to and triggered by specific lexical tokens. All actions take the form of 'macros' to provide update operations on semantic trees, instantiated as IF…THEN…ELSE rules that yield semantically transparent structures when applied (see e.g., Figure 2).

Computational Actions form a small, fixed set of macros. Some encode the properties of the lambda calculus and the logical tree formalism (the logic of finite trees; LOFT: Blackburn & Meyer-Viol, 1994): e.g., THINNING, which removes satisfied requirements; and ELIMINATION, which performs beta-reduction of a node's daughters, and annotates the mother node with the resulting formula. Other computational actions reflect the fundamental predictivity and dynamics of DS, e.g., COMPLETION, which moves the pointer up and out of a sub-tree once all requirements therein are satisfied; and ANTICIPATION, which moves the pointer from a mother node to a daughter node with any unfulfilled requirements. While the former set of actions are *inferential*, thus not adding any new information to the trees, the latter set introduce alternative parse paths, thus capturing structural ambiguity: COMPLETION, for example, precludes any further development of the current sub-tree because it moves the pointer up and out of it. In general, computational actions can apply optionally whenever their preconditions are met, but are not triggered by lexical input. The successful parse of a word w_1 thus amounts to finding a sequence of computational actions (possibly empty) that leads to a tree that satisfies the preconditions of the lexical action for w_1. The parse search process/history can be represented as a Directed Acyclic Graph (DAG), with (partial) semantic trees as nodes, and actions as edges, i.e., transitions between trees.

Lexical actions are associated with word forms in a DS lexicon. Like computational actions, these are tree-update macros composed of sequences of atomic actions that are basic tree-building operations such as `make`, `put` and `go`. `Make` creates a new (daughter) node, `go` moves the pointer, and `put` decorates the pointed node with a node label. Figure 2 shows an example for a proper noun, *John*. The action checks whether the pointed node (marked as $\Diamond$) has a requirement for type *e*; if so, it decorates it with type *e* (thus satisfying the requirement), the semantics for *John* (see Section 2.3 for details) and the bottom restriction $\langle\downarrow\rangle\bot$ (meaning that the node cannot have any daughters). Otherwise (if there is no requirement $?Ty(e)$), the action aborts, meaning that the word '*John*' cannot be parsed in the context of the current tree.

2.3. Type Theory with Records (TTR)

Previous efforts have incorporated TTR (Cooper, 2005, 2012) as the semantic formalism in which meaning representations are couched in DS (Eshghi et al., 2012; Purver et al., 2011, 2010)—it is within the DS-TTR fusion that we express our model.

TTR is an extension of standard type theory, and has been shown to be useful in contextual and semantic modelling in dialogue (see, e.g., Fernández, 2006; Ginzburg, 2012; Purver et al., 2010, among many others).

With its rich notions of underspecification and subtyping, TTR has proved crucial for DS research in the incremental specification of content (Hough, 2015; Purver et al., 2011); in the specification of a richer notion of dialogue context (Purver et al., 2010); in models for learning dialogue systems from data (Eshghi & Lemon, 2014; Eshghi et al., 2017; Kalatzis et al., 2016); and, as we employ it here, in models of DS grammar learning (Eshghi et al., 2013a, 2013b) and in integrating linguistic with perceptual semantics (Dobnik et al., 2012; Larsson, 2013; Yu et al., 2016).

In TTR, logical forms are specified as *record types*, which are sequences of *fields* of the form $[\, l : T \,]$ containing a label l and a type T. Record types can be witnessed (i.e., judged true) by *records* of that type, where a record is a sequence of label–value pairs $[\, l = v \,]$. We say that $[\, l = v \,]$ is of type $[\, l : T \,]$ just in case v is of type T.

Fields can be *manifest*, i.e., given a singleton type, e.g., $[\, l : T_a \,]$, where T_a is the type of which only a is a member; here, we write this as $[\, l_{=a} : T \,]$. Fields can also be *dependent* on fields preceding them (i.e., higher) in the record type (see Figure 4).

$$R_1 : \begin{bmatrix} l_1 & : T_1 \\ l_{2=a} & : T_2 \\ l_{3=p(l_2)} & : T_3 \end{bmatrix} \quad R_2 : \begin{bmatrix} l_1 & : T_1 \\ l_2 & : T_{2'} \end{bmatrix} \quad R_3 : []$$

Figure 4. Example TTR record types.

The standard *subtype* relation $\sqsubseteq$ can be defined for record types: $R_1 \sqsubseteq R_2$ if for all fields $[\, l : T_2 \,]$ in R_2, R_1 contains $[\, l : T_1 \,]$ where $T_1 \sqsubseteq T_2$. In Figure 4, $R_1 \sqsubseteq R_2$ if $T_2 \sqsubseteq T_{2'}$, and both R_1 and R_2 are subtypes of R_3. This subtyping relation allows semantic information to be incrementally specified, i.e., record types can be indefinitely extended with more information and/or constraints.

We also make use of the *meet type* of two or more RTs and an operation to yield an equivalent RT to their meet type. As Cooper (2012) explains, the meet of two RTs results in a type that is no longer an RT, even if the objects it witnesses are records, however an *equivalent* RT to the meet type of two RTs R_1 and R_2 is the yield of a merge operation $(R_1 \mathbin{\barwedge} R_2)$ (Larsson, 2010). Two types T_1 and T_2 are equivalent iff for any object a in the domain such that iff $a : T_1$ then $a : T_2$ and vice-versa. In the simplest case merge can be characterised as union of fields of two RTs, for example, for R_1 and R_2 in (1). This generalises to cases of more than two RTs.

$$\begin{aligned} \text{if } R_1 = \begin{bmatrix} l_1 & : T_1 \\ l_2 & : T_2 \end{bmatrix} \quad \text{and } \; R_2 = \begin{bmatrix} l_2 & : T_2 \\ l_3 & : T_3 \end{bmatrix} \\ R_1 \wedge R_2 \equiv R_1 \mathbin{\barwedge} R_2 \quad = \begin{bmatrix} l_1 & : T_1 \\ l_2 & : T_2 \\ l_3 & : T_3 \end{bmatrix} \end{aligned} \tag{1}$$

We also make use of *asymmetric merge* to define 'meaning as state change': ($\boxed{\barwedge}$) as described by Dobnik et al. (2012). Operationally this differs from the standard merge in that given two record types R_1 and R_2, $R_1 \boxed{\barwedge} R_2$ will yield a RT which is the union of all fields with labels not shared by R_1 and R_2 and the asymmetric merge of the remaining fields with the same labels, whereby R_2's type values take priority over R_1's fields, yielding a resulting RT with R_2's fields only in those cases. A simple example is given in (2).

$$\begin{aligned} \text{if } R_1 = \begin{bmatrix} l_1 & : T_1 \\ l_2 & : T_2 \end{bmatrix} \quad \text{and } \; R_2 = \begin{bmatrix} l_2 & : T_{2'} \\ l_3 & : T_3 \end{bmatrix} \\ R_1 \boxed{\barwedge} R_2 \quad = \begin{bmatrix} l_1 & : T_1 \\ l_2 & : T_{2'} \\ l_3 & : T_3 \end{bmatrix} \end{aligned} \tag{2}$$

2.4. Grounding TTR Record Types (RT) into the BabyAI World

There is a long line of work showing how TTR can be used to integrate linguistic and perceptual semantics (see Dobnik et al., 2012; Hough et al., 2020; Larsson, 2013; Yu et al., 2016, a.o.), in other words, in grounding NL semantics into objects, events, their properties and relations between them in the real world.

In this paper, we follow the same line of work in assuming that a BabyAI agent has to be attuned to (or have access to) an ontology of basic types to make sense of its raw observations of BabyAI states. Here we closely mimic the *types-as-classifiers* approach of Hough et al. (2020) in that we take basic, atomic TTR RTs to be grounded in (potentially learnable) classifiers that are in turn applied to whole or partial BabyAI states—see Figure 5. As we will see just now, this allows the agent to compositionally construct the *maximal type* of individual BabyAI states or parts thereof (e.g., the agent's immediate vicinity) wherein all the relevant properties of the state are captured in a fully compositional manner. The

resulting maximal type therefore constitutes how the agent makes sense of or perceives its environment.

The agent's domain ontology is a set, $O = T \cup P \cup Rel$, of *basic* (atomic) RTs where T is the set of types of objects (e.g., doors, bells, keys), P is the set of their properties (e.g., colours), and Rel is the set of relations that might hold between objects in the environment. Figure 5 shows example RTs in each of T, P, and Rel.

T (types of objects)	$\begin{bmatrix} obj1 & : & Ind \\ obj1{-}type & : & door(obj1) \end{bmatrix}$	type of state with a door
P (properties of objects)	$\begin{bmatrix} obj1 & : & Ind \\ obj1{-}col & : & blue(obj1) \end{bmatrix}$	type of state with something blue
Rel (relations between objects)	$\begin{bmatrix} obj1 & : & Ind \\ agent & : & Ind \\ orient & : & facing(agent, obj1) \end{bmatrix}$	type of state where something is facing something else

Figure 5. Example atomic RTs from our ontology.

Given the ontology, O, and any raw observation of a BabyAI state, s, the agent proceeds to make a *maximal type judgement* about s by *merging* (using the merge or meet operation defined above in Section 2.3) all the relevant atomic types τ in O where $s : \tau$, i.e., by merging all the types in the set $tot = \{\tau \in O | s : \tau\}$. Thus, the agent is able to make the following maximal type (henceforth MT) judgement: $s : MT$ where $MT = \bigwedge_{\tau \in tot} \tau$.

The upshot of this process is that the agent can analyse the structure of its environment by drawing upon the basic types in its ontology to compositionally construct the maximal type of any given BabyAI (total or partial) state. Figure 6 shows examples of 2×2 raw state observations together with their maximal type thus constructed; where, e.g., in (1), the agent judges the state to be of a type where there is a box and the agent is next to this box; and in (2), of a type where there is a green door with the agent facing it (because it is facing down).

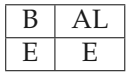
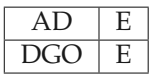
	State	Maximal Type (MT)
(1)		$\begin{bmatrix} obj1 & : & Ind \\ obj1{-}type & : & box(obj1) \\ agent & : & Ind \\ orient & : & facing(agent, obj1) \end{bmatrix}$
(2)		$\begin{bmatrix} obj1 & : & Ind \\ obj1{-}type & : & door(obj1) \\ obj1{-}col & : & green(obj1) \\ agent & : & Ind \\ orient & : & facing(agent, obj1) \\ obj1{-}open & : & open(obj1) \end{bmatrix}$

Figure 6. Examples of maximal type judgements.

This types-as-classifiers (symbol) grounding methodology, though complex, has several important benefits: (1) it is fully compositional in that it allows information from said classifiers (types) to be combined (composed) in any configuration even if such a configuration is never observed in the training data—this provides the backbone for *compositional generalisation*; (2) it allows the agent to reason, via the TTR calculus, over its perceptual experience; and, as will become clear below; (3) it models the agent's perceptual and linguistic understanding within one and the same formal language (TTR), thus allowing seamless integration between Dynamic Syntax on the one hand, and the BabyAI multi-modal en-

vironment on the other. (4) It provides a simple bridge not just to automated planning (see below), but also to, e.g., Object-Oriented Reinforcement Learning (Diuk et al., 2008).

2.5. Problem Statement: Grounded Language Learning in the BabyAI Environment

Addressing our research questions, the aim of this work is to provide and compare models for grounded language learning in the BabyAI environment, to shed more light on the data efficiency and generalisation capabilities of grammar-based versus neural architectures.

In the standard BabyAI setup, end-to-end models are trained to predict a sequence of actions that solve a mission expressed in a NL instruction (see, e.g., Figure 1); i.e., models are trained on pairs of the form $\langle inst, demo \rangle$ where *inst* is an instruction in English; and *demo* is a sequence of actions by the oracle BabyAI agent that solves that mission—the latter is what is referred to as a 'teacher demonstration'.

Modular Architecture: Contrary to all the previous research on BabyAI, instead of providing end-to-end models that predict a sequence of actions directly, we take a different, more modular approach. We tackle this problem by splitting it into two modules: (1) a Natural Language Understanding (NLU) module, whereby the instruction is semantically parsed into a TTR Record Type; and (2) Planning, by using an off-the-shelf planner to generate the actions to be taken by the agent to solve the corresponding mission, after the parsed RT is mapped to goal representations in the appropriate planning language. Since (2) is off the shelf and the mapping function to planning goals is manually specified, this modular setup means that the BabyAI grounded language learning problem is reduced to learning a TTR semantic parser for the BabyAI fragment of English. To be explicit: the only learned component in our system is the NLU module (the TTR semantic parser); the planner and the `rt2goal` mapping are fixed and receive no training signal.

2.5.1. Meaning as State Change

To derive the symbolic meaning of an instruction in TTR—and to learn different semantic parsers from it later—we take the meaning of an instruction to be the symbolic specification of *how the state of the agent's vicinity changes* after the instruction is performed, i.e., we take the meaning of the instruction to be its *perlocutionary effect*; e.g., the meaning of "go to the red box" is an RT that specifies what type of state best characterises `being at the red box`.

Given this characterisation, we are able to generate the training data needed to train and compare our semantic parsing models; namely, data of the form $\langle inst, R_{instr} \rangle$ where *inst* is an instruction and R_{instr} is an RT specifying the perlocutionary effect—or semantics—of *inst*. To capture R_{instr} as state change, we use the asymmetric merge operation in TTR, whereby: $R_{goal} = R_{init} \boxed{\wedge} R_{instr}$, where R_{init} is the initial state of the agent for the mission, and R_{goal} is the successful goal state reached by the agent after executing the teacher demonstration given by the BabyAI oracle agent. Asymmetric merge of two RTs (see above Section 2.3) is suitable here because it retains all the semantic information in the left operand, while replacing the type of its clashing labels by those of the right operand; thus capturing the change from the left hand side RT to the right-hand side RT.

The possible values of R_{instr} can therefore be drawn from the set $\{R_{instr} | R_{goal} = R_{init} \boxed{\wedge} R_{instr}\}$ for it to successfully convey the necessary state change to reach the goal. For the purposes of efficient language learning described below, we have to select one of these possible record types for the semantics of the instruction: here we make an assumption that this RT will be defined as the difference RT, $R_{goal} \setminus R_{init}$ merged with every field in R_{goal} *dependent* on (i.e., being a predicate with an argument of) any field in $R_{goal} \setminus R_{init}$—see Figure 7. This preserves the maximal amount of information which could be used in

a verbal instruction to achieve the goal. While this could have redundancy in that, for example, a speaker may not mention the colour of a target object *obj*1 and therefore the *obj*1−*col* field is not needed, for this paper, we err on the side of using the maximal possible semantics of a successful instruction as a training example.

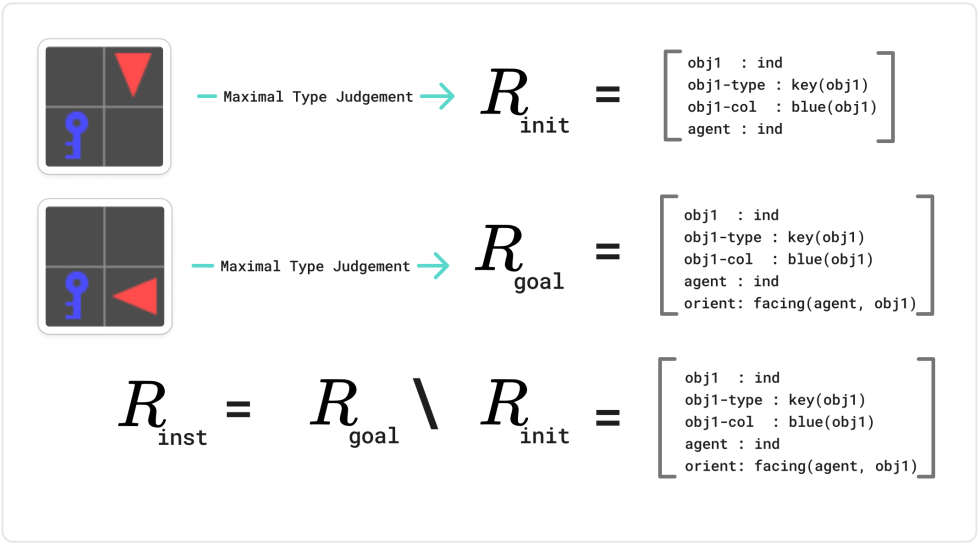


Figure 7. Meaning as state change. Example instruction: `go to the blue key`.

2.5.2. Overall Architecture

With the above in place, Figure 8 now shows the overall modularised architecture of the system where the red box is the NLU module: a learnable TTR semantic parser of BabyAI instructions, in principle replacable by any TTR parser. It is essentially the learnability, data efficiency and generalisation properties of this module that we study in this paper.

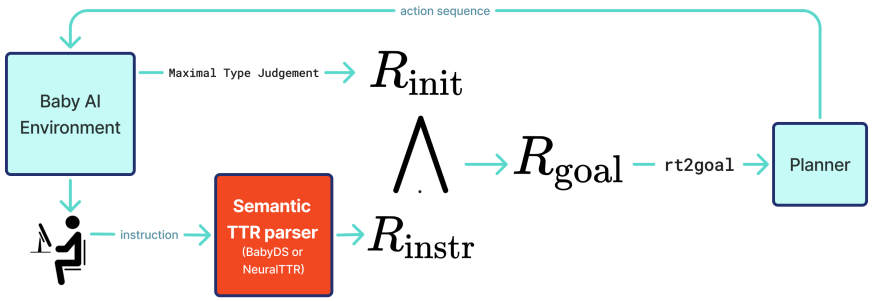


Figure 8. Modularised BabyAI architecture.

At inference time for an individual human instruction, the pipeline is as follows: the instruction is parsed by the semantic TTR parser, and the result is merged with the type judgement over the initial state of the agent, viz. R_{init}, giving us an RT that specifies the mission goal: R_{goal}. This is then converted into the input language of the planner module using the `rt2goal` function. In this instance, we have implemented the `rt2goal` function to map R_{goal} to the existing breadth-first planner goal specification in the BabyAI platform. In principle, the `Planner` module can be replaced with any planner, or indeed any compatible Reinforcement Learning model—the mapping to an Object-Oriented Reinforcement Learning reward function would likely be straightforward, though we leave this to future work.

The `Planner` then returns a sequence of actions that are run by the platform, the success of which is then evaluated, in turn contributing to the Task Success Rate (TSR) that we report below in all our extrinsic evaluation experiments.

2.6. Models

We are now set up to present the details of the models that we will be comparing in our experiments below. As we would like to compare the performance of our probabilistic model with other neural baselines in a few different ways, we train a neural model. The details of our implementation for our grammar-based model and the neural baselines are available in Sections 2.6.1 and 2.6.2, respectively.

2.6.1. BABY-DS: Grounded Grammar Induction with Continual Learning

This is learned following the algorithm and methodology of Eshghi et al. (2013a) for DS-TTR grammar induction, using a version of the Expectation–Maximisation algorithm (Dempster et al., 1977) to induce a probabilistic semantic parser from our <Instruction, TTR semantics> pairs. Supervision is provided exclusively through these pairs: no other component of the pipeline (the planner, the grounding function, or the `rt2goal` mapping) receives any training signal. The resulting BABY-DS parser operates by maintaining a set of candidate parses at each word, ranked by their probabilities.

Following Eshghi et al. (2013a), the learning algorithm induces a probabilistic DS lexicon from pairs of sentences and target meanings represented as TTR record types. Crucially, this approach requires less supervision than prior methods (Eshghi et al., 2013b): rather than assuming target logical forms have a pre-defined compositional tree structure, the algorithm must discover how to decompose a relatively flat record type into incremental, word-level contributions. While DS computational actions are treated as fixed from the outset, lexical actions for each word are learned via induction as training data is presented.

As per Algorithm 1, for each new sentence–RT pair, the algorithm proceeds in two phases. In the first phase, hypothesis lattices are constructed for each word in the sentence, consisting of DS-TTR atomic actions along with a probability for each path; these lattices encode all possible monotonic tree extensions leading from the initial axiom—the requirement for a complete proposition—to the target RT. The search over this space is constrained by a type lattice that efficiently represents all possible TTR increments required to reach the target meaning. Hypotheses are categorised into tree-building actions, which create the necessary daughter nodes through functional decomposition or type extension, and content actions, which decorate tree leaves with specific formulae drawn from the target RT; these are packaged into macros that ensure the resulting trees remain well-formed under the grammar's logic. For words not previously encountered, a hypothesis lattice is added to the model with uniform probabilities over all paths; for words already in the model, stored lattices are updated with any new paths by intersecting the two DAGs.

In the second phase, an incremental variant of the Expectation–Maximisation algorithm manages the ambiguity of assigning sub-meanings to individual words. In the Expectation step, the model enumerates all hypothesis tuples—combinations of word-level action sequences that jointly satisfy the target RT—and weights each according to current lexical probability estimates. In the Maximisation step, the probabilities of individual word–action pairs are re-estimated by summing and normalising their weighted contributions across all successful tuples for that sentence. The resulting estimates are then incorporated into the global lexicon via a weighted average over the number of sentences processed so far, enabling sentence-by-sentence refinement of the grammar without requiring reprocessing of the entire corpus—a design that mirrors the incrementality characteristic of human language acquisition.

Algorithm 1 Incremental Semantic Grammar Induction

Require: Training corpus $\mathcal{D}$, Computational actions G, Initial lexicon Θ
Ensure: Updated probabilistic lexicon $\Theta = \{\theta_w\}_{w \in \mathcal{V}}$
  for all sentence-target pairs $\langle S, T_t \rangle \in \mathcal{D}$ **do**
    {Stage 1: Hypothesis Construction & Splitting}
    $\mathcal{P} \leftarrow$ Search for monotonic action sequences from axiom $T_0 \rightarrow T_t$
    $HT \leftarrow$ Split each path $\in \mathcal{P}$ into tuples $\langle h_1, \ldots, h_n \rangle$ where each h_i is a word hypothesis s.t.:
    - h_i contains exactly one content decoration action.
    - h_i is computationally maximal on the left.

    $\mathcal{H}_S \leftarrow$ Generalise tuples into equivalence classes via intersection.
    {Stage 2: Incremental Expectation–Maximisation Probability Estimation}
    for all Hypothesis Tuples $HT_j = \langle h_{j1}, \ldots, h_{jn} \rangle \in \mathcal{H}_S$ **do**
      $p(HT_j|S) \leftarrow \prod_{i=1}^{n} \theta'_{w_i}(h_i^j)$ {E-step: Expectation}
    end for
    for all words $w \in S$ **do**
      $\theta''_w(h) \leftarrow \frac{1}{s} Z \sum_{j:h \in HT_j} p(HT_j|S)$ {M-step: Re-estimation}
      $\theta_w^N(h) \leftarrow \frac{N-1}{N} \theta_w^{N-1}(h) + \frac{1}{N} \theta''_w(h)$ {Incremental update}
      Handle unseen h by redistributing mass from known hypotheses.
    end for
  end for

The model trained and evaluated for addressing RQ1 (see Section 2.8.1) follows the same procedure as the original work, but in RQ2 (see Section 2.8.2), we address a major limitation of this algorithm: online curriculum learning for better generalisation and robust handling of unknown words. We provide the details of this below.

In all the work on DS grammar induction so far (including the setup used for RQ1 below), the learner has had a limitation where it could not exploit previously learned information for more efficient learning of only unknown words. This means that in a scenario where some of the words in a training sample are known and some have not been seen before, it has not been possible to learn only the new words and parse the familiar words based on what was learned previously. Therefore all the words in the training sample would be treated as unknown and all had to be hypothesised from scratch. This also means we had a computationally inefficient learning algorithm, since the search space for word hypotheses grows exponentially with the number of words in the sample, effectively making learning from samples containing at least 7 words almost impossible.

In this work, this limitation is addressed and its drastic impact in terms of learning efficiency has become apparent: A class 2 instruction, after tokenisation, can have maximum seven words (e.g., "putnextto the red ball a green box". Prior to this, the learner of Eshghi et al. (2013a) would have to search among $2^7 = 128$ hypotheses to be able to come up with consistent lexical actions for all the words in the sample, and we could see hypothesising lexical actions over a training set of 40 such samples simply could not terminate.

To solve this constraining issue, we have implemented into the learner the ability to "parse and learn"; when encountering a training sample, the words that are known to the model (the ones available in a previously learned model that is provided as the base lexicon) are parsed and the rest (unknown words) are hypothesised irrespective of their position in the sentence. These hypotheses for the unknown words are further filtered based on the context; there has to be consistency between how a newly hypothesised lexical action is connecting the semantics of the preceding and succeeding words (if any).

This addition effectively enables curriculum learning in DS induction. We can now exploit the knowledge learned from linguistically simpler data (in our case, class 1) and be

able to generalise to more complex data (as can be found in Section 3.2) in a considerably more optimal way by reducing the number of lexical hypotheses exponentially. For example, in learning of the above sample, instead of 128 hypotheses to search among, since our model now does not have to hypothesise the whole sentence but only "putnextto" in a way that is compatible with the lexicon learned from class 1 and used for parsing the rest of the sentence, we can witness there is only one hypothesis available for "putnextto" and therefore the corresponding lexical action is seamlessly learned from a single training sample.

The improvement it brings to the induction algorithm is studied in Section 2.8.2 and the results in terms of data efficiency are available in Section 3.2.

2.6.2. Neural LSTM Parser

In this paper, we compare our BABY-DS parser to a simple neural baseline: a sequence-to-sequence model (Sutskever et al., 2014) based on the LSTM architecture (Hochreiter & Schmidhuber, 1997). We use this model for neural semantic parsing; it therefore plays exactly the same role as the BABY-DS parser in our modular architecture (see Figure 8) and receives precisely the same supervised ⟨Instruction, TTR semantics⟩ training pairs for principled comparison. Crucially, the LSTM model both learns from and produces TTR record types: during training it receives linearised TTR record types as target sequences, and at inference time it generates TTR record types that are evaluated using the same metrics as BABY-DS, ensuring a fully like-for-like comparison. Although LSTM-based models are not considered state-of-the-art architectures any more (after the introduction of Transformers by Vaswani et al. (2017)), we choose this architecture because it works incrementally: it can receive partial inputs and yield partial outputs as words of the instruction are being received on the fly.

For the sake of our experiments and getting the best results from our neural models, we perform hyper-parameter tuning (on both class 1 and class 2 data as explained in Section 2.8.1), the details of which can be found in Appendix A. For Experiment 2 (as per Section 2.8.2), we pick the same hyper-parameters that give the best results for class 2 in RQ1. Furthermore, we adapt teacher-forcing for learning stability.

2.7. Data Generation for Semantic Parsing

In Section 2.1, we explained our choice of environment for grounded language learning. As our focus is mainly language learning, we first define our notion of *mission classes* based on the linguistic complexity of the instruction of the BabyAI levels. These classes are: (1) The family of GoTo-PickUp-Open instructions (the simplest class). (2) PutNextTo (a separate class, as there is a reference to two objects as opposed to class 1 containing one object in the instruction). (3) 2-subgoals missions and (4) 4-subgoals missions. In this paper, as proof-of-concept baselines, we continue with class 1 and 2 of BabyAI missions as our data and leave classes 3–4 for future work. Example instructions from classes 1–2 are as below:

class 1: go to the box
open the green door
class 2: put a yellow ball next to the blue key
put the purple ball next to the yellow box

We generate the data based on all the combinatorial compositions that the BabyAI BNF grammar can produce, as pairs of a sentence (mission instructions) and semantics (meaning of the instruction represented in TTR), which, as illustrated by Figure 7, is characterised as a TTR representation of the state change caused by the agent successfully following the instruction. These pairs are then fed into our models as per Section 2.6.1 (and how it has been used by Eshghi et al. (2013a)) for grammar learning.

For example, to obtain the dataset for class 1, all the possible combinations of the verbs (go, open, pick up), nouns (ball, box, key, door), articles (a, the) and adjectives (red, green, blue, purple, yellow, grey, and none), giving a total of 112 samples, are generated accompanied by their target TTR semantics.

Tokenisation: Since each token in DS corresponds to exactly one lexical action, multi-word expressions that function as a single semantic unit must be treated as single tokens. Accordingly, we concatenate multi-word verbs in our instructions prior to parsing: "go to" becomes `goto`, "pick up" becomes `pickup` and "put ... next to" becomes `putnextto` (with the intervening object descriptions remaining as separate tokens). This tokenisation keeps the number of tokens per instruction low—at most 5 for class 1 and 7 for class 2—which is important because the hypothesis search space during DS grammar induction grows exponentially with the number of tokens in a training sample. The same tokenisation is applied consistently across all models to ensure a fair comparison.

Since neural models require vector inputs and cannot directly learn from TTR record types, we convert target TTR record types to flattened representations and use an embedding layer at the beginning of our architecture to learn a numerical representation for the elements in a record type. This flattening is fully reversible, so we can reconstruct proper TTR record types from the model's outputs without loss of information for the evaluation step.

2.8. Experiments

In this research, our focus is not only grounded language learning, but also a model that is able to process the input incrementally, and that is data-efficient and can generalise well from simpler scenarios to more complex ones. Given this focus, we investigate the performance of our models in two ways: (1) a general intrinsic and extrinsic evaluation, and (2) generalisation and catastrophic forgetting of the models. We provide the details of how we perform each of these tests below in their respective sections. Table 2 summarises the key differences between the two experiments.

Table 2. Comparison of the experimental setup for Experiment 1 and Experiment 2.

	Experiment 1: Data Efficiency	**Experiment 2: Generalisation & Forgetting**
Goal	Measure data efficiency	Measure generalisation and catastrophic forgetting
Source data	Class 1 (112)/Class 2 (624)	Class 1 (112) as source
Target data	Same class	Class 2 (2352) as target
Training scheme	Cumulative batches, same class	Pretrain on source, fine-tune on target
Test sets	10% held-out from same class	Generalisation (class 2) + Forgetting (class 1)
Metrics	F1, Coverage, EM, TSR	F1, Coverage, EM

2.8.1. Experiment 1 Setup: Data Efficiency

In this experiment, we conduct both intrinsic and extrinsic evaluations to assess the data efficiency of our BABY-DS parser compared to neural baselines. The core question here is: How much training data does each type of model require to reach acceptable performance?

Data: We generate the training data by exhaustively enumerating all possible instructions that the BabyAI BNF grammar can produce for our two data classes (as explained in Section 2.7). For class 1, this yields 112 unique instruction–semantics pairs. For class 2, we obtain 624 pairs when restricting to maximum-one-adjective instructions. We exclude two-adjective combinations from class 2 at this stage since DS induction becomes computationally prohibitive with such long sequences as the search space grows exponentially, and in a fair setup, training data between the models should be equal.

From class 1, we reserve 10% of the data as a test set and another 10% as a development set (to control for early stopping), leaving 80% for training. The training portion is then

divided into cumulative batches, where each batch contains an additional 10% of the full dataset. So, batch 1 contains the first 10%, batch 2 contains 20%, and so on. This cumulative design, where each model is always trained on all data seen so far, lets us construct learning curves that directly measure data efficiency: a model that reaches high performance with fewer cumulative samples is deemed more data-efficient. Concretely, for class 1, each successive batch adds approximately 11 new samples (10% of the 112-sample pool), yielding a test set of size 13. For class 2, the batch increment is set to approximately 12 samples per step (from the 624-sample pool), keeping the per-step growth comparable across classes.

Models & Training: The training and inference of BABY-DS follows the general procedure explained in Section 2.6.1. At inference time, we consider the top-N most probable parses (with N $\in$ {1, 2, 3}) and evaluate each variant separately. This gives us insight into whether increasing the beam size helps performance, and by how much.

For the neural baselines, we first perform hyper-parameter tuning, the details of which are available in Appendix A for class 1 data and class 2 data separately (on the full training split, i.e., the final cumulative batch containing 80% of the data), and then pick the first and second best performing models (NTR-L & NTR-S as referred to in Section 3.1, respectively). We then train each of these models on the cumulative batches and monitor their performance on the development set to trigger early stopping if necessary. For BABY-DS, early stopping is triggered when the product of coverage and exact match (Coverage $\times$ EM; both defined below) reaches a threshold of 0.8. For the neural models, training is stopped when validation accuracy reaches 0.994 or when no improvement is observed for five consecutive epochs (patience = 5), whichever comes first.

Evaluation: Finally, for the intrinsic evaluation, we record the performance of our models on the same test set for all batches and report the semantic accuracy of our parsers. We measure this using three metrics. Coverage (parse coverage) is the percentage of test sentences for which the system obtains a complete parse, measuring the grammar's ability to provide an interpretation for the input. F1-score measures semantic accuracy from precision and recall, computed by comparing induced record types to gold logical forms via field alignment (maxMapping, following Eshghi et al. (2013a)). Exact Match (EM; same-formula accuracy) is the percentage of parses that produce a logical form identical to the gold standard, assessed here through two-way TTR subsumption checking. This is done after training on each of the cumulative batches, to then represent data efficiency; a model that converges faster, meaning that it gives good performance with less data, is the desired one. At the end, we report the average over five runs with seeded shuffling of the training data for more realistic results.

It is noteworthy that although the definition of "coverage" for BABY-DS is straightforward, this is not as clear for the neural models, requiring its own definition here as it differs from the term used in the context of BABY-DS: because neural networks always generate an output as the result (parsing in our case), one might think the coverage is always 100%, but it should be modified to mean "if the model can make a valid TTR record type" so that this ratio is made more meaningful.

In terms of the extrinsic evaluation, we put the model in action (as is demonstrated in Figure 8) and report the Task Success Rate for both of our models. We generate a seeded set of 50 missions that fall into our two classes of data, use our parsers to specify a goal state (explained in Section 2.5), use BabyAI's `verify` methods to confirm if the agent ends up in the goal state of the mission or not, and report the ratio of success to total number of trials.

2.8.2. Experiment 2 Setup: Generalisation and Forgetting

It has long been known that many neural models struggle with systematic generalisation—the ability to transfer the knowledge from previous learning from potentially simpler tasks to more complex, but partially overlapping scenarios, and requiring less data compared to starting learning on the more complex target task from scratch. Moreover, it has been observed among this family of models that they tend to forget their knowledge of the source data when trained on unseen, target data. In other words, their performance drops significantly if tested back on the source data when fine-tuned on the target data (McClelland et al., 1995). Therefore in this section, we design a set of experiments to study the generalisation and forgetting of our models to see if these hold for our parsing task.

Our general experimental setup is to train our models, BABY-DS and a neural baseline, on the data from a source class (in our case, class 1 data) that is linguistically simpler, and to continue fine-tuning these models on a more complex, target class (in our case here, class 2), to measure how fast and efficiently the models learn to show good performance on the target class, and by how much starting training on the source class boosts learning on the target data. Simultaneously, we test these models on a test set from the source class to measure if and how much models would forget what they have previously learned from the source class. Details of these steps are explained below.

Data: The data for this experiment are generated in the same fashion as explained in Section 2.7 and Experiment 1. Therefore we have 112 samples for class 1 data; for class 2, we use the full data that can be generated by the BabyAI grammar, so a total of 2352 samples are available (note that in Experiment 1, this was maximum-one-adjective). It follows that class 1 is chosen as our source class and class 2 as our target class for our experiment. Subsequently, two sets composed of 10% of class 2 data are picked as the generalisation set and development set (to look for early stopping), respectively. Additionally, 10% of class 1 data is set aside, to measure the forgetting of our models on.

For building our training data to measure generalisation and forgetting, we follow the idea of feeding our model with cumulative batches of data similar to Experiment 1. Here, we start with a batch of class 1 data that has given the best results in Experiment 1 for each model, and in the following batches, add 150 samples from class 2 data. However, the batch sizes necessarily differ between the two model families. Because BABY-DS converges with far fewer samples (as shown in Section 3.2), providing it with the same large batches as the neural model would introduce redundant data without affecting outcomes. Concretely, for the neural baseline, each cumulative batch adds 150 class 2 samples, whereas for BABY-DS each batch adds only one additional sample beyond the previous batch. This asymmetry reflects the models' differing data requirements rather than an unfair advantage.

Models & Training: Our competing models here again are BABY-DS with learned class 1 models provided as seed lexicon to it, and similar to Experiment 1, a neural LSTM-based model that we call NTR-GEN, with the same hyper-parameters as the model trained on class 2 data in Experiment 1, and trained on the batches of data described above.

Evaluation: We test BABY-DS and the trained neural model after each batch on the same generalisation and forgetting sets. The results are reported in terms of F1-Score, Coverage and Exact Match (EM) similar to Experiment 1, and then five-fold cross-validation averages are reported to give a more realistic insight (similar to what is done in Experiment 1—see Section 3.1). We finally make the following comparisons: (I) BABY-DS and NTR-GEN gains in terms of EM on generalisation and forgetting testing sets and the amount of data consumed by the models for these results, as well as (II) for each of our models, comparing their EM on the generalisation test set when trained on only class 2 data and when having

started training on class 1 data and further fine-tuning on class 2 to reveal the effect of curriculum learning.

3. Results

3.1. Experiment 1 Results: Data Efficiency

Having followed the steps in Section 2.8.1, we get the results in Figure 9 for class 1 and Figure 10 for class 2.

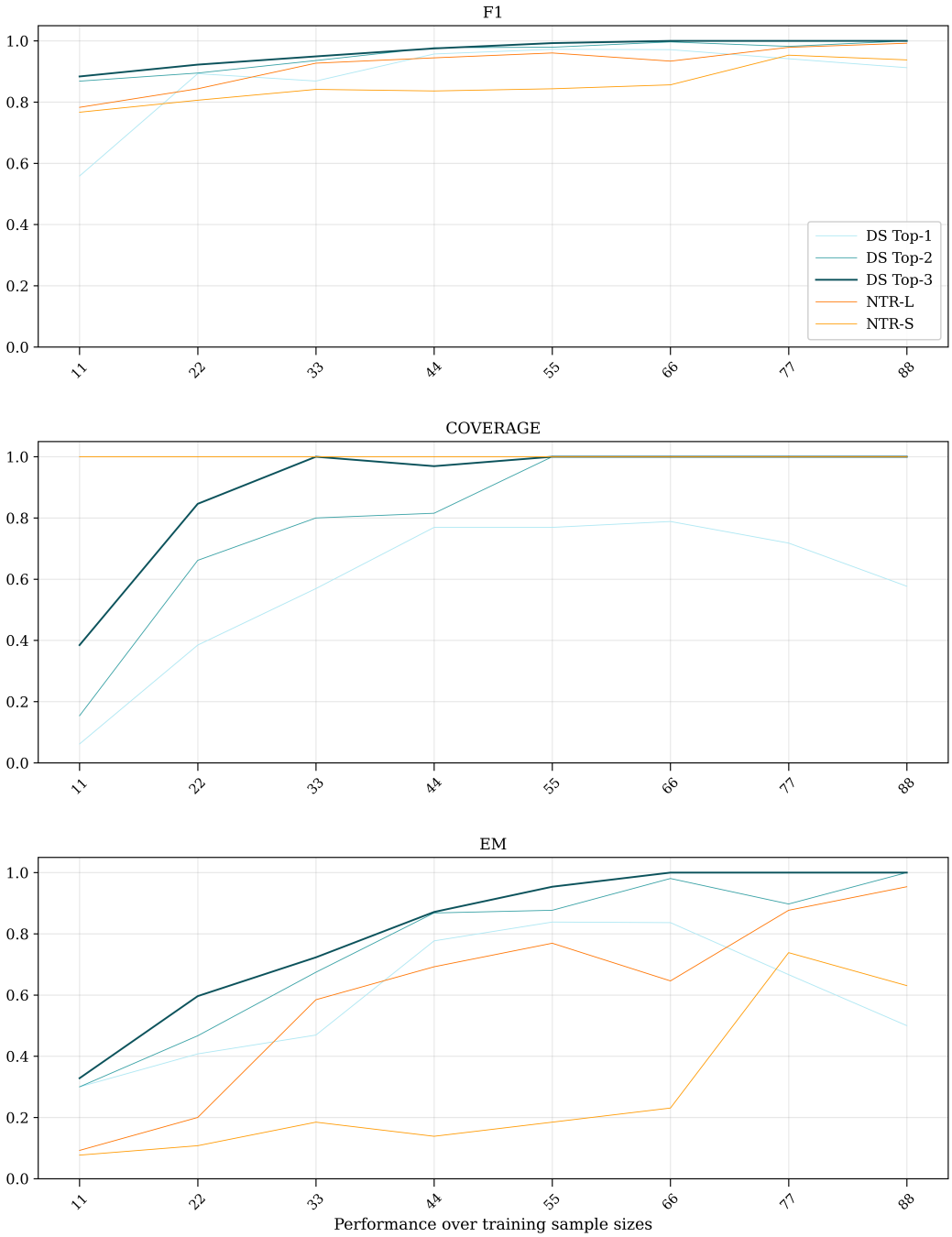


Figure 9. BABY-DS and neural competitor results on class 1 for data efficiency. Performance measures are plotted against cumulative batch training size.

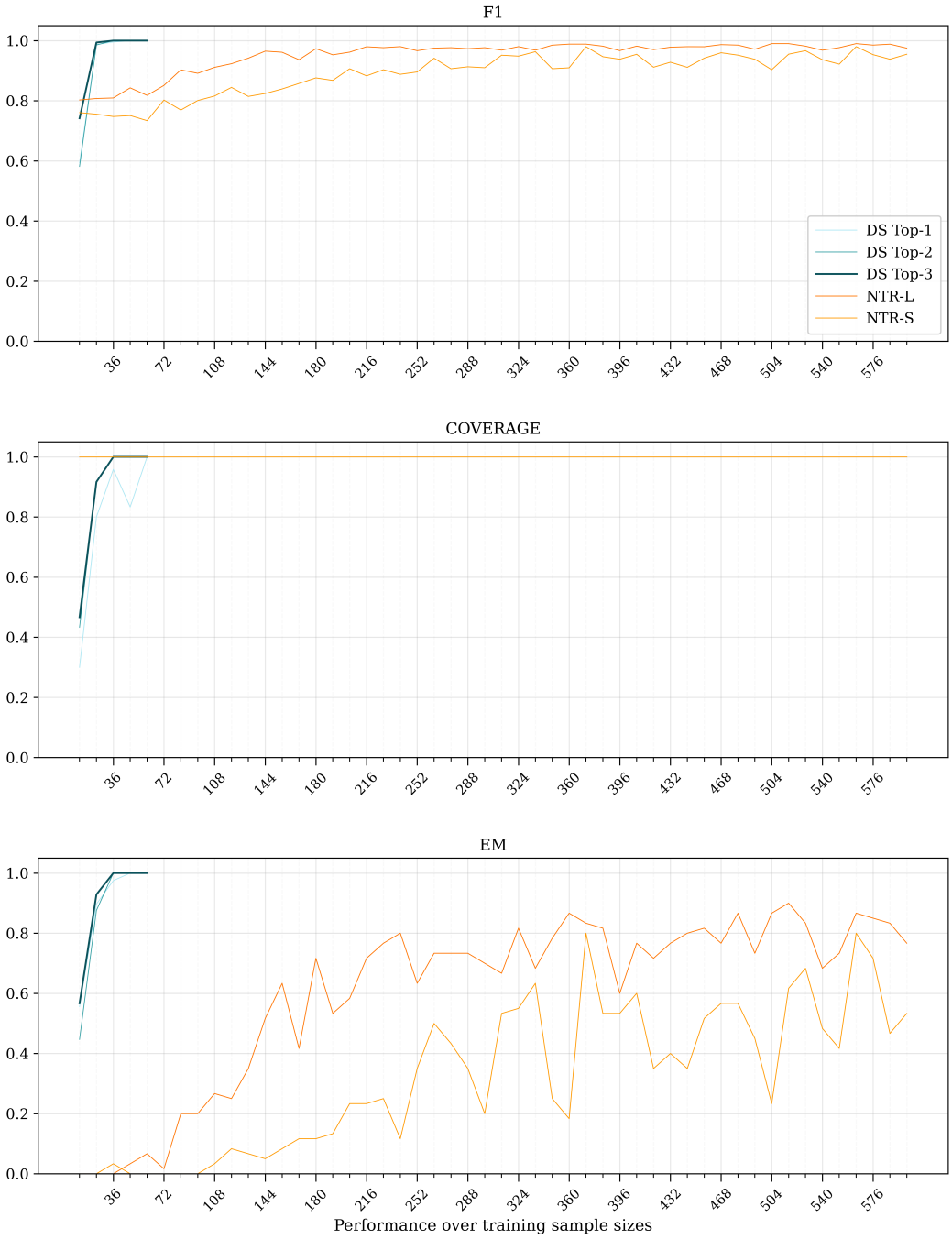


Figure 10. BABY-DS and neural competitor results on class 2 for data efficiency.

As can be seen, all three BABY-DS models, top-1, top-2, and top-3, converge substantially faster than the neural baselines on both data classes. The top-1 model (a single best parse) reaches high performance on class 1 after 6 batches (66 samples) and shows strong results on class 2 as well. The top-2 model, which considers the two highest-ranked candidate parses, further improves upon top-1 by achieving faster convergence and higher final performance on both classes. The top-3 model achieves a perfect result on class 1 after 6 batches (66 samples), with class 2 converging even faster at only 3 batches (36 samples). On the other hand, our larger neural baseline shows near-perfect performance after 9 batches (99 samples) on class 1, and after 20 batches (360 samples) on class 2.

We observe that BABY-DS converges more rapidly in class 1 and much more visibly on class 2, compared to the neural competitors, demonstrating superior data efficiency on both data classes while maintaining greater stability (given the fluctuations in NTR-L results).

For the fairest comparison, we focus on the BABY-DS top-1 results against NTR-L. Even at top-1, BABY-DS converges faster: it reaches high performance after 6 batches on class 1 and shows strong results on class 2 as well. Top-2, which expands the beam to the two best candidate parses, provides a meaningful improvement over top-1, achieving faster convergence and higher scores without requiring any additional training data. The top-3 induced lexicon further extends this trend, achieving a perfect model on class 1 after 6 batches (66 samples), and on class 2 after only 3 batches (36 samples). This progression reflects the fact that BABY-DS maintains a ranked set of candidate parses; a comparable top-N exploration strategy could in principle be implemented for the LSTM baseline (e.g., via beam search), but we have not done so in this work.

With regards to our neural models, since the smaller neural model (NTR-S, with almost 600k parameters) shows highly unstable behaviour compared to the larger model (NTR-L with almost 1m parameters), we only use the latter for our Experiment 2.

For our extrinsic evaluation of Task Success Rate (TSR), we test our BABY-DS models (top-1, top-2, top-3) and NTR-L on 50 actual BabyAI environments (as explained in Section 2.8.1) and get the results as shown in Table 3.

Table 3. Experiment 1 Task Success Rate (TSR) for the trained agents.

Experiment	Model	TSR
class 1	Top-1	0.82
	Top-2	1.00
	Top-3	1.00
	NTR	0.94
class 2	Top-1	1.00
	Top-2	1.00
	Top-3	1.00
	NTR	0.90

It is observable here that these results are exactly the same as the EM score reported in our intrinsic evaluation. This is due to the fact that based on our methodology and the deterministic design of the BabyAI planner, if an instruction's semantics from our parsers is correct, this is mapped to the TTR representation of the goal state (the input to the planner) seamlessly and therefore giving the same ratio of correctness. Focusing on the primary top-1 comparison: BABY-DS already achieves a TSR of 0.82 on class 1 and a perfect 1.00 on class 2, compared to NTR's 0.94 and 0.90, respectively. Expanding to top-2 and top-3 further improves BABY-DS to perfect TSR on both classes.

It is important to note that we do not measure the independent influence of the off-the-shelf planner on TSR. What matters for our purposes is that the experimental comparisons are sound: the planner is identical across all conditions and models. Whatever bias or limitation the planner introduces in one condition is equally present in all others; therefore, differences in TSR are attributable solely to differences in the semantic parser.

3.2. Experiment 2 Results: Generalisation and Forgetting

In this section, we will see the performance from our models given the training and evaluation setup discussed previously in Section 2.8.2, to compare BABY-DS and our neural baseline, NTR-GEN.

First, the generalisation and forgetting scores of NTR-GEN along with a neural model trained only on class 2 data (NTR-C2) can be found in Figure 11.

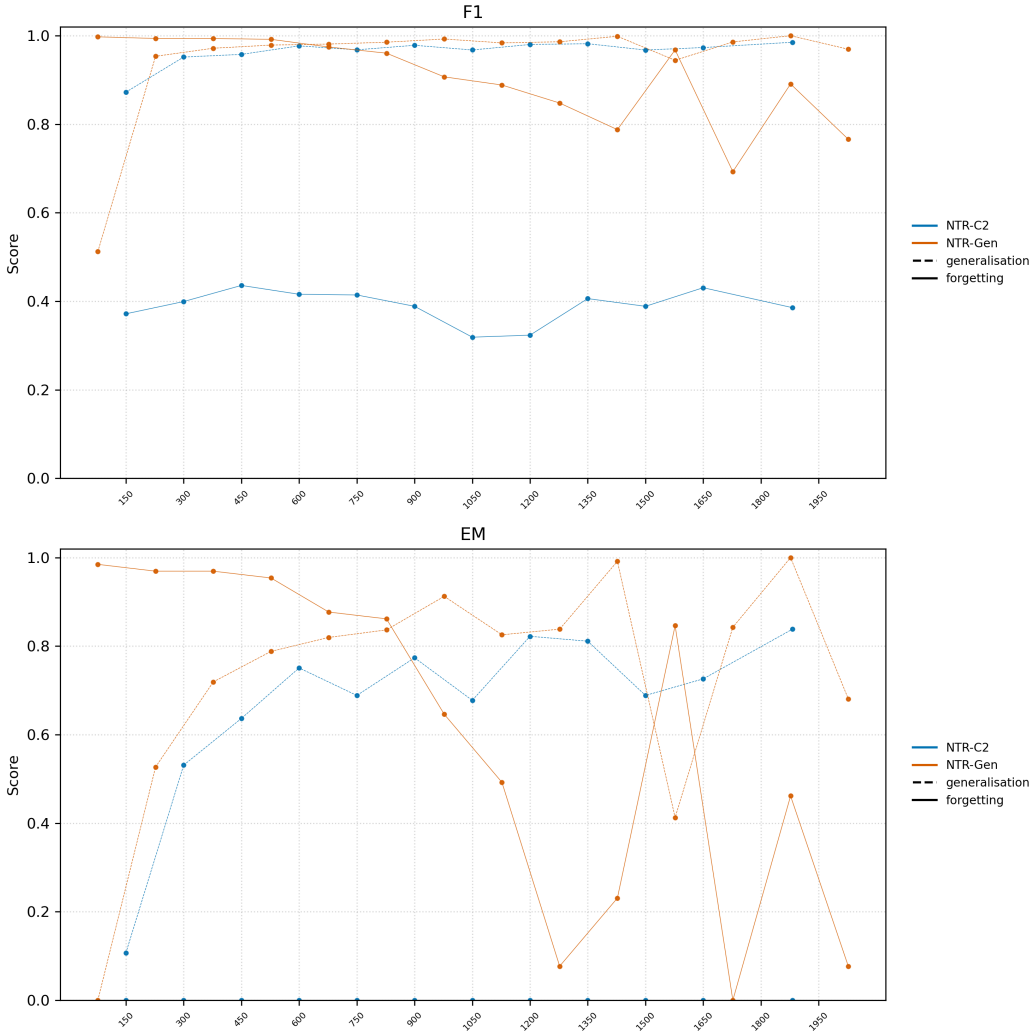


Figure 11. Generalisation and forgetting of our neural parsing baseline (NTR-GEN) from class 1 to class 2 data, coupled with the results from NTR-C2 tested on the same class 2 generalisation set to demonstrate the curriculum learning effect. Coverage score is removed here due to being ill-defined for neural models.

Overall, our results confirm our hypothesis that NTR-GEN does not generalise efficiently. As shown in Figure 11, although NTR-GEN shows normal performance on the generalisation (class 2) test set, after six batches of additional class 2 training data, the model's performance on forgetting the (class 1) test set drops considerably. This behaviour, as the term suggests, is in fact catastrophic since two recoveries from poor performance on the forgetting set can be observed in further batches. With regard to another matter, by comparing the EM score of NTR-C2 and NTR-GEN on all of the training batches, we can see that NTR-GEN sees a boost in performance from being trained on class 1 data. This confirms that curriculum learning can provide some improvements to model performance, although not significantly.

The above results from the neural baseline are, however, far from comparable to how DS efficiently learns and performs; as it is evident in Table 4, all three BABY-DS variants demonstrate strong generalisation from very few class 2 training samples, with performance improving markedly from top-1 through top-3. Most strikingly, *one training sample* is enough for the top-3 model to give near perfect generalisation scores with no forgetting.

Table 4. Generalisation and forgetting results for BABY-DS. Training Size here is the average number of training samples from class 2 required for the model to generalise. Best results are in bold.

Model	Training Size	Class 2			Class 1		
		F1	Coverage	EM	F1	Coverage	EM
Top-1	3.4	71.73	66.18	70.55	97.32	89.09	90.91
Top-2	2.6	87.85	80.36	86.55	98.51	100.00	90.91
Top-3	**1.0**	**99.84**	**100.00**	**98.18**	**99.36**	**100.00**	**96.36**

The results in Table 4 reveal a clear progression across the three BABY-DS beam configurations. Top-1 already achieves a noteworthy 70.55% EM on the class 2 generalisation set from an average of just 3.4 training samples, while retaining 90.91% EM on the class 1 forgetting set—indicating moderate but non-negligible forgetting. Top-2 improves both generalisation (86.55% EM) and data efficiency (requiring only 2.6 samples on average), with the same class 1 retention. Most strikingly, our top-3 induced lexical actions give near-perfect generalisation (98.18% EM on class 2) after being trained on a single sample from class 2, with no forgetting visible (96.36% EM on class 1). This progression demonstrates that expanding the candidate parse beam consistently improves both generalisation and data efficiency. Furthermore, this shows the power of our added interleaved parsing and induction method, where the model that could not learn from the complete class 2 data (due to computational complexity of the learning algorithm) can seamlessly benefit from curriculum learning and generalize to class 2 after being trained on a single sample, demonstrating great data efficiency in doing so.

Drawing our attention to both of our models, it is well-established that BABY-DS compared to NTR-GEN has shown superior generalisation while remaining data-efficient, given the 98.18% EM score with 78 training samples evaluated against the 1427 samples required by NTR-GEN to give 100% EM score, and does so while not forgetting previously learned information, given the 96.36% EM score on the class 1 test set by BABY-DS compared to the 22% EM score of NTR-GEN on the forgetting set where it gives perfect results on the generalisation test set.

Ultimately, the significant data efficiency and performance observed here by BABY-DS allows for building models that enable human-in-the-loop collaborative learning, itself being one of the initial motivations behind the creation of BabyAI (Chevalier-Boisvert et al., 2018).

4. Discussion

We conclude by discussing how our research questions RQ1 and RQ2 have been addressed in our two sets of experiments, before finishing with the limitations of the work and future proposed work.

In the first experiment, we observe that class 2 is learned much faster than class 1 by BABY-DS (after 66 and 36, samples respectively) which can be unexpected. But we believe class 2 requires less data (although longer in terms of the number of tokens in each sentence) compared to class 1 because of the variety of verbs—which lead the grammatical structure of the sentence—in class 1 is higher than class 2.

In our second experiment, the results given by NTR-GEN may seem poor; therefore, to confirm the correctness of our architecture and training process, we provide the model with multiple different combinations of class 1 and class 2 data, and check the performance on the same generalisation and forgetting sets. All of these results are documented in Appendix C.

It is noteworthy that in the current setup, the amount of data in the base models in the grammar-based parser seems to play an important role: DS induction generally does not

require as much data as neural models (as far as it has been tested), but also, we can see that slightly more samples in the base model give closer to perfect performance, and this is propagated further into generalisation scores. So, for example, as per our observations, it is an imperfect base model (among the five base models based on different random seeds) that causes slightly imperfect generalisation scores, and perfect base models have always given 100% EM in terms of generalisation. Furthermore, NTR-L struggles with a low number of samples in the batches and never gives perfect EM scores.

An important question concerns whether the compositionality exhibited by BABY-DS is learned from data or engineered into the framework. Compositionality is built into Dynamic Syntax itself, which standardly employs the lambda calculus over symbolic representations to derive phrase- and sentence-level meanings from individual word meanings via function application (beta-reduction). Crucially, DS is agnostic to the particular semantic representation formalism used underneath: original DS used the epsilon calculus (Cann et al., 2005; Kempson et al., 2001); subsequent work has coupled DS with RDF (Addlesee & Eshghi, 2021) and with vector space semantics (Purver et al., 2021; Sadrzadeh et al., 2018). In every case, compositionality arises from theoretical linguistic–semantic modelling rather than from data-driven learning, and it is precisely this built-in compositional bias that constitutes one of the crucial inductive biases allowing the DS-TTR model to generalise so well from limited data, as our experiments demonstrate.

4.1. Limitations

While previous research on Dynamic Syntax has explored related challenges, such as Shalyminov et al. (2017) who studied the data efficiency of DS but in a model that was not visually grounded, and Yu et al. (2017) who studied visual grounding but not data efficiency, our work bridges these gaps. Specifically, we have studied data efficiency for visually grounded DS models, as well as testing for catastrophic forgetting. Because we are tackling these aspects simultaneously using a strictly incremental semantic grammar, we have made simplifying assumptions that make our model limited in a few ways that we describe here.

Starting with the main one, the grounding function described in Section 2.4, i.e., the domain ontology and the mapping from raw BabyAI states to TTR record types via maximal type judgements, is hand-engineered rather than learned from perceptual data. This effectively reduces the task from end-to-end visually grounded language learning, where semantics would be induced from raw observations, to a supervised grammar learning setup where symbolic meaning representations are provided. In principle, following the types-as-classifiers approach we adopt from Larsson (2013), these classifiers can be learned jointly within the system; we leave this to future work.

Another limitation of our current study is the absence of evaluation on class 3 and class 4 instruction data. However, extending the model to these classes presents a promising avenue to demonstrate the efficacy of our methodology on substantially longer and compositionally more complex sequences. Based on the systematic generalisation from class 1 to class 2 that we observed in RQ2, we believe generalisation to classes 3 and 4 is possible with exposure to a small number of training samples (i.e., in a few-shot learning setting). We expect that with minimal exposure to class 3 instructions, such as "pick up the ball before you go to a purple box", the model could learn the semantics of conjunctions and sequential operators (e.g., "before", "after", and "and"). The model could then generalise to the full complexity of class 4 data, which comprises even longer sequences (i.e., four mission subgoals), without needing to learn any new vocabulary, as the lexicon remains identical to that of class 3. In contrast, due to a lack of systematic generalisation, neural models would likely require large amounts of training data to generalise to these

more complex classes, and there is a high probability they will demonstrate catastrophic forgetting—as shown here and in previous research. Addressing these challenges, alongside the interpretation of directional instructions (e.g., "go to the door behind you"), is left as future work. Nevertheless, it is important to mention that the current absence of evaluation on classes 3 and 4 does not stem from a theoretical limitation of the general learning algorithm—similar structures were learned by this algorithm previously in Eshghi et al. (2013a)—but rather from a minor lack of infrastructure in the current implementation of the BABY-DS learner. While our efforts to develop a robust end-to-end neural baseline yielded suboptimal results, this may partly reflect the inherent architectural limitations of LSTMs, or our choice of flattened TTR representations. Future investigations should therefore explore more modern recurrent architectures, such as xLSTMs (Beck et al., 2024), which have demonstrated superior long-term memory capabilities compared to alternative recurrent architectures as well as Transformer-based models, as well as the impact of more structured TTR encoding as the input to neural models.

4.2. Future Work

With the ultimate aim of scaling our model for wide-coverage parsing, drawing inspiration from neuro-symbolic approaches to wide-coverage grounded language learning (Eshghi et al., 2022; Kogkalidis, 2023; Lindemann, 2025; Prange, 2022; Soulos, 2025), we aim to scale our models beyond previous work (Mao et al., 2021; Yu et al., 2017) to more complex environments and closer to the real world. On the architectural level, a hybrid of probabilistic neural learning (McCoy & Griffiths, 2025; Ponti et al., 2019) or generally focusing more on the learning of structure rather than deep learning architecture engineering could be a fruitful direction. Related to this, and with a view to improving the data efficiency of the current work, one promising technique is softly injecting linguistic inductive bias into models (e.g., Nandi et al. (2024) among others) for more accurate, data-efficient parsing.

In the current setting, the learning agent only receives the semantics of the instruction in order to take actions in the environment, meaning that the agent does not interact with the oracle or a human-in-the-loop teacher. Adding the capability of asking clarification questions or providing completions (e.g., following Eshghi and Ashrafzadeh (2023); Hough (2015) for incremental language generation by parsing) on the agent side, while receiving feedback from an oracle to boost performance (McCallum et al., 2025, 2023) and potentially increased data efficiency on the other, could also be explored in future work in order to make the interaction more naturalistic.

Closely related to the interactive setting described above, while incremental processing is a core strength of DS, evaluating real-time language understanding—including incremental parsing behaviour of our models following the metrics introduced by Baumann et al. (2011), as well as real-time reactions to instructions in the environment as they are partially received and before the instruction has been fully uttered, similar to human dialogue—has not been conducted and is left as an interesting path forward.

Finally, a longer-term direction concerns the perceptual grounding of our models. A natural further investigation would compare them with pre-trained Visual Language Models, possibly by extending BABY-DS with a vision module such as that proposed by Matsson et al. (2019). Taken together, pursuing these directions puts us on a path towards developing more cognitively plausible models (Oh & Linzen, 2025).

Author Contributions: Conceptualization, A.A., J.H. and A.E.; methodology, A.A., J.H. and A.E.; software, A.A. and A.E.; validation, A.A., J.H. and A.E.; formal analysis, A.A., J.H. and A.E.; investigation, A.A., J.H. and A.E.; resources, J.H. and A.E.; data curation, A.A.; writing—original draft preparation, A.A., J.H. and A.E.; writing—review and editing, A.A., J.H. and A.E.; visualization, A.A., J.H. and

A.E.; supervision, J.H. and A.E.; project administration, A.A., J.H. and A.E.; funding acquisition, A.A., J.H. and A.E. All authors have read and agreed to the published version of the manuscript.

Funding: Hough's work is supported by the EPSRC grant EP/X009343/1 'FLUIDITY'.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The code, trained models, and data used in this paper are publicly available at https://github.com/Dynamics-of-Language, accessed on 12 April 2026.

Acknowledgments: The authors would like to thank Sabrina McCallum, Alessandro Suglia, Carl Bettosi, and Andy Edmondson for helpful discussions and valuable feedback on earlier versions of this work. We also thank the anonymous reviewers for their constructive comments, which helped improve the manuscript. During the preparation of this manuscript/study, the authors used Cursor and Claude Opus 4.6 for the purposes of proofreading and assisting in drafting portions of the experimental code. The authors have reviewed and edited the output and take full responsibility for the content of this publication.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

BNF	Backus–Naur Form
CCG	Combinatory Categorial Grammar
CG	Compositional Generalisation
CL	Curriculum Learning
DAG	Directed Acyclic Graph
DS	Dynamic Syntax
DS-TTR	Dynamic Syntax and Type Theory with Records
EAGER	Asking and Answering Questions for Automatic Reward Shaping in Language-guided RL
ELLA	Exploration through Learned Language Abstraction
EM	Exact Match
F1	F1-score (harmonic mean of precision and recall)
FiLM	Feature-wise Linear Modulation
FOSSIL	Feedback on Suboptimal Samples for Imitation Learning
G2L2	Grammar-Based Grounded Lexicon Learning
GLAM	Grounded LAnguage Models
GRU	Gated Recurrent Unit
HPT	Hyper-Parameter Tuning
IL	Imitation Learning
LAST	Language-guided Skill Learning with Temporal Variational Inference
LLM	Large Language Model
LOFT	Logic of Finite Trees
LSTM	Long Short-Term Memory Network
MemN2N	Memory Networks (End-to-End)
MT	Maximal Type
NL	Natural Language
NLU	Natural Language Understanding
NTR	Neural TTR (parser)
OL	Online Learning
PDDL	Planning Domain Definition Language
PPO	Proximal Policy Optimisation
QA	Question Answering
RDF	Resource Description Framework

RL	Reinforcement Learning
RT	Record Type
RQ1	Research Question 1 (data efficiency)
RQ2	Research Question 2 (generalisation and forgetting)
SA	Semantic Accuracy
SE	Sample Efficiency
SOTA	State of the Art
TSR	Task Success Rate
TTR	Type Theory with Records
VLM	Vision–Language Model

Appendix A

Here, we provide details of the hyper-parameter tuning that has been performed for our neural baselines on class 1 and class 2 data in Experiment 1. Given the architecture design described in Section 2.6.2, we first take 10% of class 1 and class 2 data as the development set and another 10% as the testing set, and use the rest as training. We then perform a brute-force search among following hyper-parameters of our neural model and track the accuracy of each to find the best performing ones among all and pick the top two best performing as NTR-L and NTR-S in Experiment 1.

- embedding dimension: [128, 256]
- hidden layers dimension: [128, 256]
- number of layers: [1, 2]
- learning rates: [0.005, 0.01]
- dropout rate: [0.2]
- batch size: [16, 32, 64]
- number of epochs: [70, 120]

The winning combination for NTR-L for class 1 was:

- embedding dimension: 128
- hidden layers dimension: 256
- number of layers: 1
- learning rates: 0.005
- batch size: 32
- number of epochs: 70

The second best performing model (chosen as NTR-S) was the same as the above but with hidden layers dimension equal to 128.

Finally, the best performing combination for for class 2 was:

- embedding dimension: 256
- hidden layers dimension: 256
- number of layers: 1
- learning rates: 0.005
- batch size: 64
- number of epochs: 120

Appendix B

Due to the large fluctuations in the performance scores of our neural models in Experiment 1, we decided to confirm that our base models for class 2 data are in fact correct and the fluctuations are normal. We do this by tweaking a single property of the model during training and comparing the effect of it in terms of EM score against all other conditions that we test for. These are listed below:

In Figure A1, NTR-L is the baseline and is the same as the model trained on class 2 data in Experiment 1.

For two cases, NTR-LB models, we used larger sizes of batches from 12 (similar to the baseline) to 30 and also 60 (called NTR-LB05 and NTR-LB10, respectively), and we could see that this change increased accuracy and gave more stable learning curves. This is a valid observation because neural models are known to be data-hungry, so if the cumulative batch size is increased, we expect to see better performance. Although, we would rather have the batch sizes remain small, since our focus has been data efficiency, and since we are comparing the performance with BABY-DS, we need to use the same amount of data for a fair comparison.

As another condition, we tested using more seeds (NTR-MS) to see if averaging over five seeds is insufficient (too small number of experiments), so we tested with 15 seeds (compared to the baseline of 5 seeds); this gave a smoother curve, but with almost the same performance as the baseline.

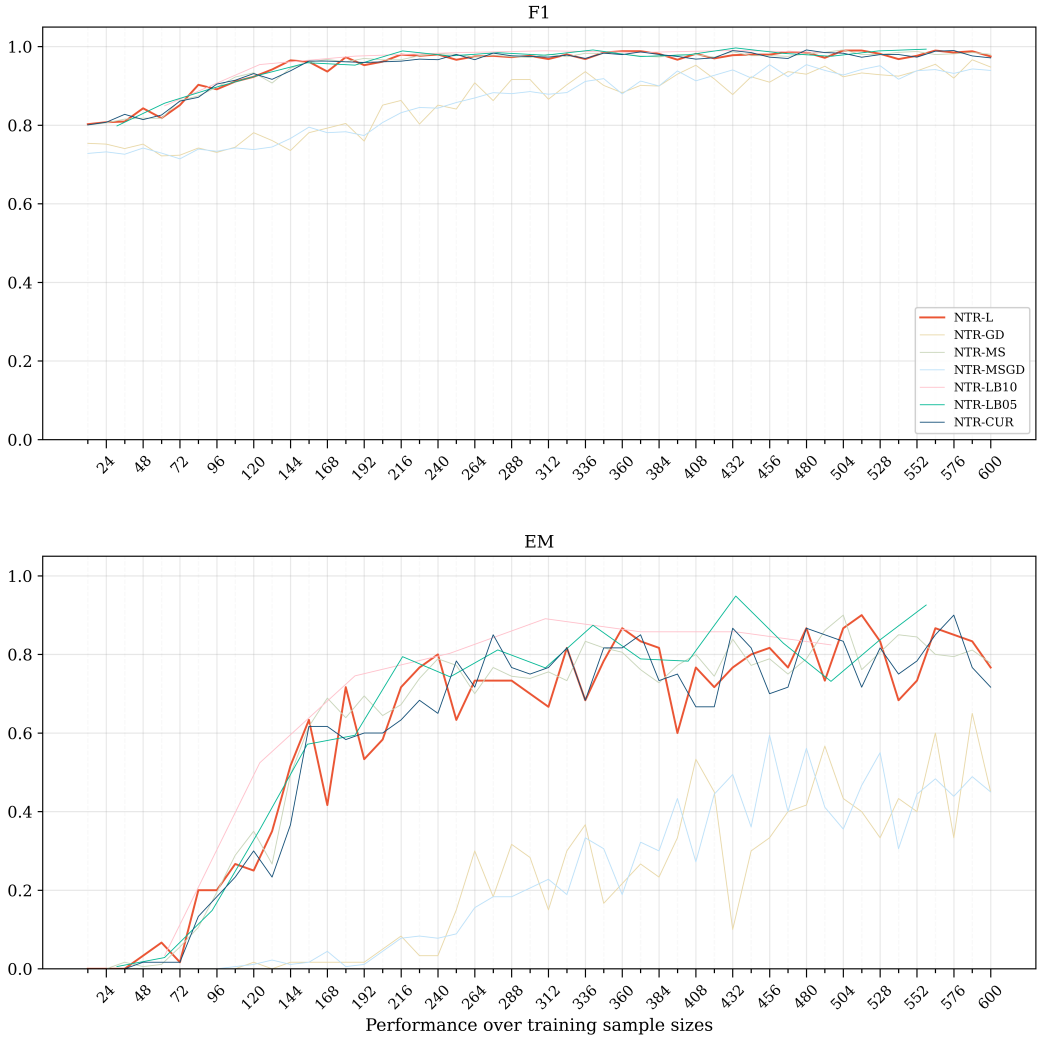


Figure A1. Confirmation of correction of architecture of the neural models in Experiment 1 for class 2 data.

We also tested for the effect of gradient decay (more specifically, exponential decay with a gamma of 0.95), with one case without increasing the number of seeds (NTR-GD) and another case with the number of seeds being 15 (NTR-MSGD). In both cases, gradient decay resulted in poor performance, with the NTR-MS model showing more stable scores (as expected). As yet another test, in NTR-CUR, we sorted the training data based on

instruction length (as a representative of the linguistic complexity of the instructions). Although it resulted in a slightly smoother training curve compared to NTR-L, it does not add much improvement towards our goal of data-efficient learning.

From these results, we conclude that the observed fluctuations in our baseline for class 2 in Experiment 1 are natural, and are a part of how the model is learning.

Appendix C

In this section, we conduct a sanity check to verify the correctness of our model used for Experiment 2. Given the low scores the model achieves on the forgetting set and the sudden fluctuations, we would like to make sure if and by how much, providing different combinations of class 1 and class 2 data to the model during training, can affect generalisation and forgetting.

To this end, we use the same experimental setup as explained in Section 2.8.2 and only modify the order and size of class 1 or class 2 data in the batches that we feed to the model. These are described below.

DATA: Here, we explain all the data combinations that we presented to the neural model to be able to pick the best among them (the colours and the order of the models mentioned here match the ones on the legend of the plots in Section 3.2).
NTR-Mixed: class 1 and class 2 data are mixed from the beginning. So, the training data includes these proportionally in cumulative batches of size 150.
NTR-C2: training data consist of cumulative batches of size 150 drawn exclusively from class 2. This condition is included to enable a controlled comparison using class 2 data alone, while keeping the batch sizes consistent with the other conditions in this experiment. As observed in Experiment 1 and in the auxiliary experiments in Appendix B, neural model performance generally improves with more training data at the model and data scales considered here.
NTR-Halves: starts with 77 samples of class 1 data as batch 1, and then we add 150 batches of class 1 and class 2 data, one in between each other, so each batch has an equal number of class 1 and class 2 samples. Given that class 2 data is generally much larger in size, when class 1 data run out, we start re-adding samples from the beginning of the shuffled class 1 data.
NTR-Proportional: starts with 77 class 1 data for the first batch, then we add 150 batches of class 1 and class 2, but the number of samples from each class in each batch is proportional to the full data size (112 samples of class 1, to 2352 samples of class 2) without repetition for class 1.
NTR-Second: This is the same as NTR-GEN in Experiment 2. Training batches start with 77 class 1 data as batch 1, then we add 150 batches of only class 2 data. This configuration matches a standard generalisation scenario in which the model first learns from simpler data before being fine-tuned on the target class.
NTR-Separate: starts with 77 class 1 data as batch 1, then we add 150 batches of alternating class 1 and class 2 data in separate blocks. That is, batch 2 adds 150 class 1 samples to batch 1, and the following batch adds 150 class 2 samples—each batch consists entirely of one class, alternating between the two.

Generalisation and forgetting scores are reported on the same data for all models. The generalisation and forgetting test sets contain 10% of each of class 2 and class 1 data, respectively (viz. 235 and 11). A development set is also picked from class 2 data with the same size, and this is used to control for early stopping of our models.

RESULTS: From the results in Figure A2 we conclude that:

NTR-Mixed: Although the performance on class 1 is very poor, some class 1 data added to training improves generalisation (comparing with NTR-C2, where the only difference is the addition of class 1 data proportionally in training batches).
We believe the low scores on the class 1 test set is generally due to the low amount of class 1 data present in training, with class 2 forming the majority of training data. Moreover, in comparison to NTR-Proportional, it can be understood that starting from a set of class 1 data is vital, as much better results on the class 1 test set are gained compared to this case where class 1 data are distributed over the whole training set.
NTR-C2: Generalisation seems normal and roughly follows the same patterns as the ones observed with a smaller batch size in RQ1 (not the smoothest and with fluctuations—see Figure 10). A direct comparison, however, cannot be made, since the data there are maximum-one-adjective class 2 data (624 samples in total) and are different from the full class 2 data here (2352 samples in total). As expected, we obtain no performance on class 1 data, and this is because the model has never seen any class 1 data (mainly verbs).
NTR-Halves: Generalisation and forgetting both seem normal, and give relatively more stable performance given the smoother curves. The great performance on the class 1 test data shows that at least in our task, the model has to be constantly reminded of the simpler data to not forget it.
NTR-Proportional: On the one hand, generalisation seems normal, with more consistent EM compared to NTR-Second. This follows a pattern we observed earlier when comparing NTR-Mixed and NTR-C2: even gradual addition of class 1 data makes generalisation smoother.
On the other hand, catastrophic forgetting is visible after the addition of five batches; we believe this is because class 2 makes up the majority of the data in the batches, and the low ratio of class 1 data (six or seven samples in a batch of 150) is not helping much with retaining performance on class 1 (this is also why the numbers in the first few batches are very close to NTR-Second).
Given the big fluctuations in the performance of this model on both test sets, it cannot be considered the most reliable in terms of generalisation and forgetting, despite being the most promising.
It is also noticeable that the forgetting of NTR-Proportional is slightly less severe (given the fluctuations) compared to NTR-Second, and more severe compared to NTR-Separate. We believe this is explained by the presence of a few class 1 samples in training over the batches for the former, but not enough samples compared to the latter.
NTR-Second: Generalisation of this model is relatively smooth, except for a sudden big drop towards the end, but generally, we can see that starting with a class 1 batch at the beginning helped with better generalisation compared to NTR-C2 overall.
The EM curve on class 1 provides us with the expected insight—the most severe catastrophic forgetting is visible for this model—it can remember class 1 up until the sixth batch, and major drops and recoveries take place.
NTR-Separate: Generalisation is quite stable, but forgetting has a massive drop and recovery towards the end.

Given the results, the best models, the ones that yield promising generalisation performance and maintain good EM on class 1, are generally those that more frequently see class 1 data during training, meaning NTR-Halves and NTR-Separate. Among these two, the most robust is NTR-Halves, as it yields the most stable generalisation and curves and higher scores.

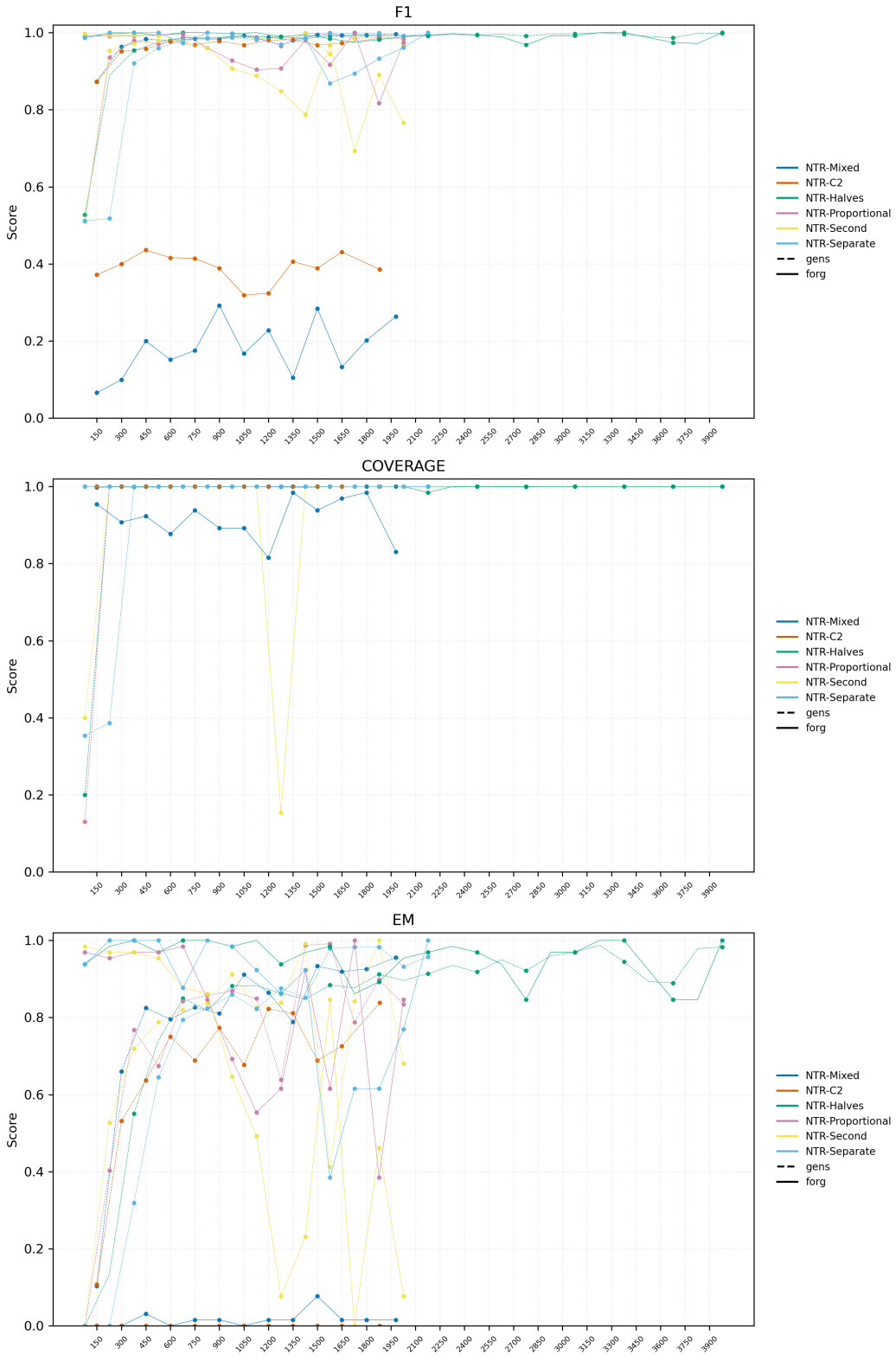


Figure A2. Generalisation and forgetting of neural parsing baselines from class 1 to class 2 data on BabyAI instructions.

References

Addlesee, A., & Eshghi, A. (2021). Incremental graph-based semantics and reasoning for conversational AI. In *Proceedings of the reasoning and interaction conference (ReInAct 2021)*. Association for Computational Linguistics.

Artzi, Y., & Zettlemoyer, L. (2013). Weakly supervised learning of semantic parsers for mapping instructions to actions. *Transactions of the Association for Computational Linguistics*, *1*, 49–62. [CrossRef]

Baroni, M. (2020). Linguistic generalization and compositionality in modern artificial neural networks. *Philosophical Transactions of the Royal Society B: Biological Sciences*, *375*, 20190307. [CrossRef] [PubMed]

Baumann, T., Buß, O., & Schlangen, D. (2011). Evaluation and optimisation of incremental processors. *Dialogue & Discourse*, *2*(1), 113–141. [CrossRef]

Beck, M., Pöppel, K., Spanring, M., Auer, A., Prudnikova, O., Kopp, M., Klambauer, G., Brandstetter, J., & Hochreiter, S. (2024, December 9–15). *xlstm: Extended long short-term memory.* 38th Conference on Neural Information Processing Systems (NeurIPS 2024), Vancouver, BC, Canada.

Blackburn, P., & Meyer-Viol, W. (1994). Linguistics, logic and finite trees. *Logic Journal of the Interest Group of Pure and Applied Logics*, *2*(1), 3–29. [CrossRef]

Bommasani, R., Hudson, D. A., Adeli, E., Altman, R., Arber, S., von Arx, S., Bernstein, M. S., Bohg, J., Bosselut, A., Brunskill, E., Brynjolfsson, E., Buch, S., Card, D., Castellon, R., Chatterji, N., Chen, A., Creel, K., Davis, J. Q., Demszky, D., … Liang, P. (2021). *On the opportunities and risks of foundation models.* Center for Research on Foundation Models (CRFM).

Bordes, A., Boureau, Y. L., & Weston, J. (2017, April 24–26). *Learning end-to-end goal-oriented dialog*. ICLR 2017, Toulon, France.

Cann, R., Kempson, R., & Marten, L. (2005). *The dynamics of language*. Cambridge University Press. [CrossRef]

Cann, R., Kempson, R., & Purver, M. (2007). Context and well-formedness: The dynamics of ellipsis. *Research on Language and Computation*, *5*(3), 333–358. [CrossRef]

Carta, T., Oudeyer, P.-Y., Sigaud, O., & Lamprier, S. (2022, December 2). *EAGER: Asking and answering questions for automatic reward shaping in language-guided RL*. NeurIPS 2022 Workshop on Language and Reinforcement Learning, New Orleans, LA, USA.

Carta, T., Romac, C., Wolf, T., Lamprier, S., Sigaud, O., & Oudeyer, P.-Y. (2023). Grounding large language models in interactive environments with online reinforcement learning. In A. Krause, E. Brunskill, K. Cho, B. Engelhardt, S. Sabato, & J. Scarlett (Eds.), *Proceedings of the 40th international conference on machine learning* (Vol. 202, pp. 3676–3713). PMLR.

Chen, D. (2012). Fast online lexicon learning for grounded language acquisition. In H. Li, C.-Y. Lin, M. Osborne, G. G. Lee, & J. C. Park (Eds.), *Proceedings of the 50th annual meeting of the association for computational linguistics* (Vol. 1: Long papers, pp. 430–439). Association for Computational Linguistics.

Chen, W., Vergari, A., & Zhao, H. (2026). Can vlms reason robustly? A neuro-symbolic investigation. *arXiv*, arXiv:2603.23867.

Chevalier-Boisvert, M., Bahdanau, D., Lahlou, S., Willems, L., Saharia, C., Nguyen, T. H., & Bengio, Y. (2018, April 30–May 3). BabyAI: A platform to study the sample efficiency of grounded language learning. International Conference on Learning Representations, Vancouver, BC, Canada.

Cooper, R. (2005). Records and record types in semantic theory. *Journal of Logic and Computation*, *15*(2), 99–112. [CrossRef]

Cooper, R. (2012). Type theory and semantics in flux. In R. Kempson, N. Asher, & T. Fernando (Eds.), *Handbook of the philosophy of science* (Vol. 14: Philosophy of Linguistics, pp. 271–323). North Holland.

Dempster, A., Laird, N., & Rubin, D. B. (1977). Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, *39*(1), 1–38. [CrossRef]

Diuk, C., Cohen, A., & Littman, M. L. (2008). An object-oriented representation for efficient reinforcement learning. In *Proceedings of the 25th international conference on machine learning* (pp. 240–247). Association for Computing Machinery. [CrossRef]

Dobnik, S., Cooper, R., & Larsson, S. (2012). Modelling language, action, and perception in type theory with records. In *Proceedings of the 7th international workshop on constraint solving and language processing (CSLP 2012)* (pp. 51–63). Springer.

Eshghi, A., & Ashrafzadeh, A. (2023, August 16–17). *Learning to generate and corr- uh I mean repair language in real-time*. SemDial 2023, Maribor, Slovenia.

Eshghi, A., Hough, J., & Purver, M. (2013a). Incremental grammar induction from child-directed dialogue utterances. In *Proceedings of the 4th annual workshop on cognitive modeling and computational linguistics (CMCL)* (pp. 94–103). Association for Computational Linguistics.

Eshghi, A., Hough, J., Purver, M., Kempson, R., & Gregoromichelaki, E. (2012). Conversational interactions: Capturing dialogue dynamics. In S. Larsson, & L., Borin (Eds.), *From quantification to conversation: Festschrift for robin cooper on the occasion of his 65th birthday* (Vol. 19, pp. 325–349). College Publications.

Eshghi, A., Howes, C., & Gregoromichelaki, E. (2022). Action coordination and learning in dialogue. In J.-P. Bernardy, R. Blanck, S. Chatzikyriakidis, S. Lappin, & A. Maskharashvili (Eds.), *Probabilistic approaches to linguistic theory* (pp. 361–422). CSLI Publications.

Eshghi, A., & Lemon, O. (2014, September 1–3). *How domain-general can we be? Learning incremental dialogue systems without dialogue acts*. SemDial 2014 (DialWatt), Edinburgh, UK.

Eshghi, A., Purver, M., Hough, J., & Sato, Y. (2013b). Probabilistic grammar induction in an incremental semantic framework. In *CSLP, lecture notes in computer science* (pp. 38–53). Springer.

Eshghi, A., Shalyminov, I., & Lemon, O. (2017). Bootstrapping incremental dialogue systems from minimal data: Linguistic knowledge or machine learning? In *Proceedings of the 2017 conference on empirical methods in natural language processing (EMNLP).* Association for Computational Linguistics.

Fernández, R. (2006). *Non-sentential utterances in dialogue: Classification, resolution and use* [Unpublished doctoral dissertation, King's College London, University of London].

Fu, H., Sharma, P., Stengel-Eskin, E., Konidaris, G., Roux, N. L., Côté, M.-A., & Yuan, X. (2024, July 21–27). *Language-guided skill learning with temporal variational inference*. ICML 2024, Vienna, Austria.

Ginzburg, J. (2012). *The interactive stance: Meaning for conversation*. Oxford University Press.

Harnad, S. (1990). The symbol grounding problem. *Physica D: Nonlinear Phenomena*, *42*(1), 335–346. [CrossRef]

Hermann, K. M., Hill, F., Green, S., Wang, F., Faulkner, R., Soyer, H., Szepesvari, D., Czarnecki, W. M., Jaderberg, M., Teplyashin, D., Wainwright, M., Apps, C., Hassabis, D., & Blunsom, P. (2017, December 4–9). *Grounded language learning in a simulated 3D world*. NeurIPS 2017, Long Beach, CA, USA.

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, *9*(8), 1735–1780. [CrossRef] [PubMed]

Hough, J. (2015). *Modelling incremental self-repair processing in dialogue* [Unpublished doctoral dissertation, Queen Mary University of London].

Hough, J., Jamone, L., Schlangen, D., Walck, G., & Haschke, R. (2020). A Types-as-classifiers approach to human-robot interaction for continuous structured state classification. In *CLASP papers in computational linguistics*. University of Gothenburg (CLASP).

Hui, D. Y.-T., Chevalier-Boisvert, M., Bahdanau, D., & Bengio, Y. (2020). BabyAI 1.1. *arXiv*, arXiv:2007.12770. [CrossRef]

Kalatzis, D., Eshghi, A., & Lemon, O. (2016). Bootstrapping incremental dialogue systems: Using linguistic knowledge to learn from minimal data. In *Proceedings of the NIPS 2016 workshop on learning methods for dialogue.* Association for Computational Linguistics.

Kempson, R., Cann, R., Gregoromichelaki, E., & Chatzikiriakidis, S. (2016). Language as Mechanisms for Interaction. *Theoretical Linguistics*, *42*(3–4), 203–275. [CrossRef]

Kempson, R., Gabbay, D., & Meyer-Viol, W. (2001). *Dynamic syntax: The flow of language understanding*. Blackwell.

Kogkalidis, K. (2023). *Dependency as modality, parsing as permutation* [Doctoral dissertation, Utrecht University]. [CrossRef]

Larsson, S. (2010). Accommodating innovative meaning in dialogue. In *Proceedings of LonDial, SemDial Workshop* (pp. 83–90). SemDial.

Larsson, S. (2013). Formal semantics for perceptual classification. *Journal of Logic and Computation*, *25*(2), 547–563. [CrossRef]

Lindemann, M. M. (2025). *Enhancing structural inductive biases of sequence-to-sequence models for semantic parsing and beyond* [Unpublished doctoral dissertation, University of Edinburgh].

Mao, J., Shi, H., Wu, J., Levy, R., & Tenenbaum, J. (2021, December 6–14). *Grammar-based grounded lexicon learning*. NeurIPS 2021, Virtual.

Matsson, A., Dobnik, S., & Larsson, S. (2019). ImageTTR: Grounding type theory with records in image classification for visual question answering. In *Proceedings of the iwcs 2019 workshop on computing semantics with types, frames and related structures* (pp. 55–64). Association for Computational Linguistics. [CrossRef]

Matuszek, C., FitzGerald, N., Zettlemoyer, L., Bo, L., & Fox, D. (2012). A joint model of language and perception for grounded attribute learning. In *ICML 2012* (pp. 1435–1442). Omnipress.

McCallum, S., Parekh, A., & Suglia, A. (2025). FOSSIL: Harnessing feedback on suboptimal samples for data-efficient generalisation with imitation learning for embodied vision-and-language tasks. In *Findings of EMNLP 2025.* Association for Computational Linguistics.

McCallum, S., Taylor-Davies, M., Albrecht, S. V., & Suglia, A. (2023). Is feedback all you need? Leveraging natural language feedback in goal-conditioned reinforcement learning. In *Workshop on goal-conditioned reinforcement learning, NeurIPS 2023.* Curran Associates Inc.

McClelland, J. L., McNaughton, B. L., & O'Reilly, R. C. (1995). Why there are complementary learning systems in the hippocampus and neocortex: Insights from the successes and failures of connectionist models of learning and memory. *Psychological Review*, *102*(3), 419–457. [CrossRef] [PubMed]

McCoy, R. T., & Griffiths, T. L. (2025). Modeling rapid language learning by distilling Bayesian priors into artificial neural networks. *Nature Communications*, *16*(1), 4676. [CrossRef] [PubMed]

Mirchandani, S., Karamcheti, S., & Sadigh, D. (2021). ELLA: Exploration through learned language abstraction. In *Proceedings of the 35th international conference on neural information processing systems.* Curran Associates Inc.

Misra, D., Langford, J., & Artzi, Y. (2017). Mapping instructions and visual observations to actions with reinforcement learning. In *EMNLP 2017* (pp. 1004–1015). Association for Computational Linguistics. [CrossRef]

Nandi, A., Manning, C. D., & Murty, S. (2024). Sneaking syntax into transformer language models with tree regularization. In *NAACL 2025* (pp. 8006–8024). Association for Computational Linguistics. [CrossRef]

Oh, B.-D., & Linzen, T. (2025). To model human linguistic prediction, make LLMs less superhuman. *arXiv*, arXiv:2510.05141.

Pantazopoulos, G., Suglia, A., & Eshghi, A. (2022). Combine to describe: Evaluating compositional generalization in image captioning. In *Proceedings of the 60th annual meeting of the association for computational linguistics: Student research workshop* (pp. 115–131). Association for Computational Linguistics. [CrossRef]

Ponti, E. M., Vulić, I., Cotterell, R., Reichart, R., & Korhonen, A. (2019). Towards zero-shot language modeling. In *Proceedings of the 2019 conference on empirical methods in natural language processing and the 9th international joint conference on natural language processing (emnlp-ijcnlp)* (pp. 2900–2910). Association for Computational Linguistics. [CrossRef]

Prange, J. (2022). *Neuro-symbolic models for constructing, comparing, and combining syntactic and semantic representations* [Unpublished doctoral dissertation, Georgetown University].

Purver, M., Eshghi, A., & Hough, J. (2011). Incremental semantic construction in a dialogue system. In J. Bos, & S. Pulman (Eds.), *Proceedings of the 9th international conference on computational semantics* (pp. 365–369). Association for Computational Linguistics.

Purver, M., Gregoromichelaki, E., Meyer-Viol, W., & Cann, R. (2010). Splitting the 'I's and crossing the 'You's: Context, speech acts and grammar. In P. Łupkowski, & M. Purver (Eds.), *Aspects of semantics and pragmatics of dialogue. Semdial 2010, 14th workshop on the semantics and pragmatics of dialogue* (pp. 43–50). Polish Society for Cognitive Science.

Purver, M., Sadrzadeh, M., Kempson, R., Wijnholds, G., & Hough, J. (2021). Incremental composition in distributional semantics. *Journal of Logic, Language and Information*, *30*, 379–406. [CrossRef]

Ross, C., Barbu, A., Berzak, Y., Myanganbayar, B., & Katz, B. (2018). Grounding language acquisition by training semantic parsers using captioned videos. In *Proceedings of the 2018 conference on empirical methods in natural language processing* (pp. 2647–2656). Association for Computational Linguistics. [CrossRef]

Sadrzadeh, M., Purver, M., & Kempson, R. (2018, April 17–18). *A tensor-based vector space semantics for dynamic syntax*. The 2nd Dynamic Syntax Conference, Edinburgh, UK.

Shalyminov, I., Eshghi, A., & Lemon, O. (2017, August 15–17). *Challenging neural dialogue models with natural data: Memory networks fail on incremental phenomena*. SemDial 2017, Saarbrücken, Germany. [CrossRef]

Soulos, P. (2025). *Unifying neural and symbolic computation for compositional generalization: Representation and processing* [Unpublished doctoral dissertation, The Johns Hopkins University].

Spilsbury, S., & Ilin, A. (2022). Compositional generalization in grounded language learning via induced model sparsity. In *Proceedings of the 2022 conference of the north american chapter of the association for computational linguistics: Human language technologies: Student research workshop* (pp. 143–155). Association for Computational Linguistics. [CrossRef]

Steedman, M. (1996). *Surface structure and interpretation*. MIT Press.

Steedman, M. (2000). *The syntactic process*. MIT Press. [CrossRef]

Suglia, A., Konstas, I., & Lemon, O. (2024). Visually grounded language learning: A review of language games, datasets, tasks, and models. *Journal of Artificial Intelligence Research*, *79*, 173–239. [CrossRef]

Sutskever, I., Vinyals, O., & Le, Q. V. (2014). Sequence to sequence learning with neural networks. In *Proceedings of the 28th international conference on neural information processing systems—Volume 2* (pp. 3104–3112). MIT Press.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., & Polosukhin, I. (2017). Attention is all you need. In *Advances in neural information processing systems* (Vol. 2017-December). Curran Associates Inc.

Wittgenstein, L. (1953). *Philosophical investigations*. Blackwell.

Yu, Y., Eshghi, A., & Lemon, O. (2016). Training an adaptive dialogue policy for interactive learning of visually grounded word meanings. In *Proceedings of sigdial 2016, 17th annual meeting of the special interest group on discourse and dialogue* (pp. 339–349). Association for Computational Linguistics.

Yu, Y., Eshghi, A., & Lemon, O. (2017). Learning how to learn: An adaptive dialogue agent for incrementally learning visually grounded word meanings. In *Proceedings of the 1st workshop on language grounding for robotics, robonlp 2017 at the 55th annual meeting of the association for computational linguistics, ACL 2017*. Association for Computational Linguistics. [CrossRef]