Bootstrapping incremental dialogue systems from minimal data: the generalisation power of dialogue grammars

Arash Eshghi  
Interaction Lab  
Heriot-Watt University  
a.eshghi@hw.ac.uk

Igor Shalyminov  
Interaction Lab  
Heriot-Watt University  
is33@hw.ac.uk

Oliver Lemon  
Interaction Lab  
Heriot-Watt University  
o.lemon@hw.ac.uk

Abstract

We investigate an end-to-end method for automatically inducing task-based dialogue systems from small amounts of unannotated dialogue data. It combines an incremental semantic grammar - Dynamic Syntax and Type Theory with Records (DS-TTR) - with Reinforcement Learning (RL), where language generation and dialogue management are a joint decision problem. The systems thus produced are incremental: dialogues are processed word-by-word, shown previously to be essential in supporting natural, spontaneous dialogue. We hypothesised that the rich linguistic knowledge within the grammar should enable a combinatorially large number of dialogue variations to be processed, even when trained on very few dialogues. Our experiments show that our model can process 74% of the Facebook AI bAbI dataset even when trained on only 0.13% of the data (5 dialogues). It can in addition process 65% of bAbI+, a corpus we created by systematically adding incremental dialogue phenomena such as restarts and self-corrections to bAbI. We compare our model with a state-of-the-art retrieval model, 2 (Bordes et al., 2017). We find that, in terms of semantic accuracy, 2 shows very poor robustness to the bAbI+ transformations even when trained on the full bAbI dataset.

1 Introduction

There are currently several key problems for the practical data-driven (rather than hand-crafted) development of task-oriented dialogue systems, among them: (1) large amounts of dialogue data are needed, i.e. thousands of examples in a domain; (2) this data is usually required to be annotated with task-specific semantic/pragmatic information for the domain (e.g. various dialogue act schemes); and (3) the resulting systems are generally turn-based, and so do not support natural spontaneous dialogue which is processed incrementally, word-by-word, with many characteristic phenomena that arise from this incrementality.

In overcoming issue (2), a recent advance made in research on (non-task) chat dialogues has been the development of so-called “end-to-end” systems, in which all components are trained from textual dialogue examples, e.g. (Sordoni et al., 2015; Vinyals and Le, 2015). However, as Bordes and Weston (2017) argued, these end-to-end methods may not transfer well to task-based settings (where the user is trying to achieve a domain goal, such as booking a flight or finding a restaurant, resulting in an API call). Bordes and Weston (2017) then presented an end-to-end method using Memory Networks (2 s), which achieves 100% performance on a test-set of 1000 dialogues, after being trained on 1000 dialogues. This method processes dialogues turn-by-turn, and so does not have the advantages of more natural incremental systems (Aist et al., 2007; Skantze and Hjalmarsson, 2010); nor does it really perform language generation, rather it’s based on a retrieval model that selects from a set of candidate system responses seen in the data.

This paper investigates an approach to these challenges - dubbed babble - using an incremental, semantic parser and generator for dialogue (Eshghi et al., 2011; Eshghi, 2015), based around the Dynamic Syntax grammar formalism (DS, Kempson et al. (2001); Cann et al. (2005)). Our advance in this paper, for end-to-end systems, is therefore twofold: (a) the method

overcomes the requirement for large amounts of dialogue data (i.e. 1000s of dialogues in a domain); (b) resulting systems are word-by-word incremental, in both parsing, generation and dialogue management. We show that using only 5 example dialogues from the bAbI Task 1 dataset (i.e. 0.13% of the training data used by (Bordes et al., 2017)) can automatically induce dialogue systems which process 74% of the bAbI testset in an incremental manner. We then introduce an extended incremental version of the bAbI dataset, which we call bAbI+ (see section 4.1), which adds some characteristic incremental phenomena - such as mid-utterance self-corrections - to the bAbI dialogues (this new dataset is freely available). Using this, we demonstrate that the system can in addition generalise to, and process 65% of the bAbI+ dataset, still when trained only on 5 dialogues from bAbI. We compare this method to (Bordes et al., 2017)’s 2, which, in terms of semantic accuracy (reflected in how well api−calls are predicted at the end of bAbI Task 1), shows very poor robustness to the bAbI+ transformations, even when it is trained on the full bAbI dataset.

This overall method is portable to other task-based domains. Furthermore, as we use a semantic parser, the semantic/contextual representations of the dialogue can be used directly for large-scale inference, required in more complex tasks (e.g. interactive QA and search).

1.1 Dimensions of Pragmatic Synonymy

There are two important dimensions along which dialogues can vary, but nevertheless, lead to identical contexts: interactional, and lexical. Interactional synonymy is analogous to syntactic synonymy - when two distinct sentences are parsed to identical logical forms - except that it occurs not only at the level of a single sentence, but at the dialogue or discourse level. Fig. 1 shows examples of interactional variants that lead to very similar final contexts, in this case, that the user wants to buy an LG phone. These dialogues can be said to be pragmatically synonymous for this domain. Arguably, a good computational model of dialogue processing, and interactional dynamics should be able to capture this synonymy.

Lexical synonymy relations, on the other hand, hold among utterances, or dialogues, when different words (or sequences of words) express meanings that are sufficiently similar in a particular domain. What is striking about lexical synonymy relations is that unlike syntactic/interactional ones, they can often break down when one moves to another domain: lexical synonymy relations are domain specific.

Eshghi & Lemon (2014) developed a method similar in spirit to Kwiatkowski et al. (2013) for capturing lexical synonymy relations by creating clusters of semantic representations based on observations that they give rise to similar or identical extra-linguistic actions observed within a domain (e.g. a data-base query, a flight booking, or any API call). Distributional methods could also be used for this purpose (see e.g. Lewis & Steedman (2013)). In general, this kind of clustering is achieved when the domain-general semantics resulting from semantic parsing is grounded in a particular domain.

We note that while interactional synonymy relations in dialogue should be accounted for by semantic grammars or formal models of dialogue structure (such as DS-TTR (Eshghi et al., 2012), or KoS (Ginzburg, 2012)), lexical synonymy relations have to be learned from data.

2 Why a grammar-based approach?

Recent end-to-end data-driven machine learning approaches treat dialogue as a sequence-to-sequence generation problem, and train their models from large datasets e.g. (Sordoni et al., 2015; Wen et al., 2016b,a; Vinyals and Le, 2015). The systems resulting from these types of approach are in principle able to handle variations/patterns that they have encountered (sufficiently often) in the training data, but not beyond.

This large-data constraint is problematic for developers but is also strange when we consider the structural knowledge that we have about language and dialogue that can be encoded in grammars and computational models of interaction. Indeed, it is often stated that for humans to learn how to perform adequately in a domain, one example is enough from which to learn (e.g. Li et. al (2006)).

Furthermore, as these systems do not parse to logical forms, they do not allow for explicit inference, which further limits their application.

We therefore develop a method combining learning from data with an incremental semantic grammar of dialogue that is able to generalise from small number of observations in a domain –
in fact even from just a few examples of successful dialogues – to a large range of interactional and syntactic variations, including everyday natural incremental phenomena.

3 Inducing Dialogue Systems

Our overall method involves incrementally parsing dialogues, and encoding the resulting semantics as state vectors in a Markov Decision Process (MDP), which is then used for Reinforcement Learning (RL) of word-level actions for system output (i.e. a combined incremental DM and NLG module for the resulting dialogue system).

3.1 Dynamic Syntax and Type Theory with Records (DS-TTR)

Dynamic Syntax (DS) is an action-based, word-by-word incremental and semantic grammar formalism (Kempson et al., 2001; Camb et al., 2005), especially suited to the highly fragmentary and context-dependent nature of dialogue. In DS, words are conditional actions - semantic updates; and dialogue is modelled as the interactive and incremental construction of contextual and semantic representations (Eshghi et al., 2015) - see Fig. 2. The contextual representations afforded by DS are of the fine-grained semantic content that is jointly negotiated/agreed upon by the interlocutors, as a result of processing questions and answers, clarification interaction, acceptances, self-other-corrections, restarts, and other characteristic incremental phenomena in dialogue - see 3 for a sketch of how self-corrections and restarts are processed via a backtrack and search mechanism over the parse search graph (see Hough (2011); Hough and Purver (2014); Eshghi et al. (2015) for details of the model, and how this parse search graph is effectively the context of the conversation). Generation/linearisation in DS is defined using trial-and-error parsing (see Section 3.2, with the provision of a generation goal, viz. the semantics of the utterance to be generated. Generation thus proceeds, just as with parsing, on a word-by-word basis (see Purver et al. (2014); Hough (2015) for details). The upshot of this is that using DS, we can not only track the semantic content of some current turn as it is being constructed (parsed or generated) word-by-word, but also the context of the conversation as whole, with the latter also encoding the grounded/agreed content of the conversation (see e.g. Fig. 4, and see Eshghi et al. (2015); Purver et al. (2010) for details). Crucially for our model below, the inherent incrementality of DS-TTR together with the word-level, as well as cross-turn, parsing constraints it provides, enables the word-by-word exploration of the space of grammatical dialogues, and the semantic and contextual representations that result from them.

Type Theory with Records (TTR) is an extension of standard type theory shown to be useful in semantics and dialogue modelling (Cooper, 2005; Ginzburg, 2012). To accommodate dialogue processing, and allow for richer representations of the dialogue context recent work has integrated DS and the TTR framework to replace the logical formalism in which meanings are expressed (Purver et al., 2010, 2011; Eshghi et al., 2012). In TTR, logical forms are specified as record types (RTs), sequences of fields of the form \([l : T]\) containing a label \(l\) and a type \(T\). RTs can be witnessed (i.e. judged as true) by records of that type, where a record is a sequence of label-value pairs \([l = v]\), and \([l = v]\) is of type \([l : T]\) just in case \(v\) is of type \(T\) (see Fig. 2 for example record types).

Importantly for us here, the standard subtype relation \(\sqsubseteq\) can be defined for record types: \(R_1 \sqsubseteq R_2\) if for all fields \([l : T_1]\) in \(R_1\), \(R_1\) contains \([l : T_1]\) where \(T_1 \subseteq T_2\). A record type can thus be indefinitely extended, and is therefore always underspecified by definition. This allows for incrementally growing meanings to be expressed in a natural way as more words are parsed or generated in

<table>
<thead>
<tr>
<th>Interactional Variation</th>
<th>USR: I would like an LG laptop sorry uhm phone</th>
<th>USR: I would like a phone by LG.</th>
<th>SYS: what would you like?</th>
</tr>
</thead>
<tbody>
<tr>
<td>SYS: okay.</td>
<td></td>
<td>SYS: sorry a what?</td>
<td></td>
</tr>
<tr>
<td>USR: a phone</td>
<td></td>
<td>SYS: by what brand?</td>
<td></td>
</tr>
<tr>
<td>SYS: by which brand?</td>
<td></td>
<td>USR: okay</td>
<td></td>
</tr>
<tr>
<td>USR: no, a phone.</td>
<td></td>
<td>SYS: by what brand?</td>
<td></td>
</tr>
<tr>
<td>SYS: okay. by which brand?</td>
<td></td>
<td>SYS: okay</td>
<td></td>
</tr>
</tbody>
</table>

Figure 1: Some Interactional Variations in a Shopping Domain
In this section we describe our method for combining incremental dialogue parsing with Reinforcement Learning for Dialogue Management (DM) and Natural Language Generation (NLG) where these are treated as a joint decision/optimisation problem.

We start with two resources: a) a DS-TTR parser DS (either learned from data (Eshghi et al., 2013a), or constructed by hand), for incremental language processing, but also, more generally, for tracking the context of the dialogue using Eshghi et al.’s model of feedback (Eshghi et al., 2015; Eshghi et al., 2011); b) a set $D$ of transcribed successful dialogues in the target domain.

We perform the following steps overall to induce a fully incremental dialogue system from $D$:

1. Automatically induce the MDP state space, $S$, and the dialogue goal, $G_D$, from $D$;
2. Automatically define the state encoding function $F : C \to S$; where $s \in S$ is a (binary) state vector, designed to extract from the current context of the dialogue, the semantic features observed in the example dialogues $D$; and $c \in C$ is a DS context, viz. a pair of TTR Record Types: $(c_p, c_g)$, where $c_p$ is the content of the current, PENDING clause as it is being constructed, but not necessarily fully grounded yet; and $c_g$ is the content already jointly built and GROUNDED by the interlocutors (loosely following the DGB model of (Ginzburg, 2012)).
3. Define the MDP action set as the DS lexicon $L$ (i.e. actions are words);
4. Define the reward function $R$ as reaching $G_D$, while minimising dialogue length.

We then solve the generated MDP using Reinforcement Learning, with a standard Q-learning method, implemented using BURLAP (McGlashan, 2016): train a policy $\pi : S \to L$, where $L$ is the DS Lexicon, and $S$ the state space induced using $F$. The system is trained in interaction with a (semantic) simulated user, also automatically built from the dialogue data and described in the next section.

The state encoding function, $F$ As shown in figure 4 the MDP state is a binary vector of size $2 \times |\Phi|$, i.e. twice the number of the RT features. The first half of the state vector contains the grounded features (i.e. agreed by the participants) $\phi_i$, while the second half contains the current semantics being incrementally built in the current dialogue utterance. Formally: $s = \langle F_1(c_p), \ldots, F_m(c_p), F_1(c_g), \ldots, F_m(c_g) \rangle$; where $F_i(c) = 1$ if $c \sqsubseteq \phi_i$, and 0 otherwise. (Recall that $\sqsubseteq$ is the RT subtype relation).

3.2.1 Semantic User Simulation

The simulator is in charge of two key tasks during training: (1) generating user turns in the right dialogue contexts; and (2) word-by-word monitoring of the utterance so far generated by the sys-
tem during exploration (i.e. babbling grammatical word sequences) by the system. To exploit (and evaluate) the full generalisation properties of the DS dialogue model, both (1) and (2) use the full machinery of the DS parser, as well the state encoding function $F$, described above. They are thus performed based on the semantic context of the dialogue so far, as tracked by DS (rather than, e.g. being based on string or template matching).

Since this includes not just the semantic features of the current turn, but also of the history of the conversation, our simulator respects the turn orderings encountered in the data, i.e. it is sensitive to the order in which information is gathered from the user.

The rules required for (1) & (2) are extracted automatically from the raw dialogue data, $D$, using $D$ and $F$. The dialogues in $D$ are parsed and encoded using $F$ incrementally. For (1), all the states that trigger the user into action, $s_i = F(c)$ – where $c$ is a DS context – immediately prior to any user turn are recorded, and mapped to what the user ends up saying in those contexts - for more than one training dialogue there may be more than one candidate (in the same context/state). The rules thus extracted will be of the form:

$$s_{	ext{trigger}} \rightarrow [u_1, \ldots , u_n],$$

where $u_i$ are user turns.

Now note that the $s_i$’s prior to the user turns also immediately follow system turns. And thus to perform (2), i.e. to monitor the system’s behaviour during training, we only need to check further that the current state resulting from processing a word generated by the system, subsumes - is extendible to - one of the $s_i$. We perform this through a simple bitmask operation (recall that the states are binary). The simulation can thus semantically identify erroneous/out-of-domain actions (words) by the system. It would then terminate the learning episode and penalise the system immediately, aiding speed of training significantly.

### 4 Evaluation

We have so far induced two prototype dialogue systems, one in an ‘electronics shopping’ domain (see Kalatzis et al. 2016) and Fig. 1) and another in a ‘restaurant-search’ domain, showing that fully incremental dialogue systems can be automatically induced from small amounts of unannotated dialogue transcripts (Kalatzis et al., 2016; Eshghi et al., 2017) - in this case both systems were bootstrapped from a single successful example dialogue. We are in the process of evaluating these systems with real users.

In this paper, however, our focus is not on building dialogue systems per se, but on: (1) studying and quantifying the interactional and structural generalisation power of the DS-TTR grammar formalism (see Section 2), and that of symbolic, grammar-based approaches to language processing more generally. We focus here on specific dialogue phenomena, such as mid-sentence self-corrections, hesitations, and restarts (see below); (2) doing the same for Bordes and Weston’s 2017 state-of-the-art, bottom up response retrieval model, without use of linguistic knowledge of any form; and (3) comparing (1) and (2).

In order to test and quantify the interactional

<table>
<thead>
<tr>
<th>Grounded Semantics</th>
<th>Current Turn Semantics</th>
<th>Dialogue so far</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x_2 : e$</td>
<td>$e_2 : es$</td>
<td>SYS: What would you like?</td>
</tr>
<tr>
<td>$e_{2 : like} : es$</td>
<td>$x_1 : USR$</td>
<td>USR: a phone</td>
</tr>
<tr>
<td>$x_{1 : USR} : e$</td>
<td>$p_{2 : prior(2)} : t$</td>
<td>SYS: by which brand?</td>
</tr>
<tr>
<td>$p_{2 : prior(2)} : t$</td>
<td>$p_{5 : sub}(2, x_1) : t$</td>
<td></td>
</tr>
<tr>
<td>$p_{5 : sub}(2, x_1) : t$</td>
<td>$p_{4 : ob}(2, x_2) : t$</td>
<td></td>
</tr>
<tr>
<td>$p_{4 : ob}(2, x_2) : t$</td>
<td>$p_{1 : phone(2)} : t$</td>
<td></td>
</tr>
<tr>
<td>$p_{1 : phone(2)} : t$</td>
<td>$x_3 : e$</td>
<td></td>
</tr>
<tr>
<td>$p_{10 : end}(2, x_3) : t$</td>
<td>$p_{9 : brand}(x_3) : t$</td>
<td></td>
</tr>
<tr>
<td>$p_{10 : end}(2, x_3) : t$</td>
<td>$p_{6 : sub}(x_3, x_5) : t$</td>
<td></td>
</tr>
</tbody>
</table>

Figure 4: Semantics to MDP state encoding with RT features
and structural generalisation power/robustness of
the two models, and 2, we need contrasting dialogue data-sets that control for interac-
tional vs. lexical variations in the input dia-
logues. Furthermore, to make our results compara-
table to the existing approach of Bordes and We-
ston (2017), we need to use the same dataset that
they have used. We therefore use Facebook AI
Research’s bAbI dialogue tasks dataset (Bordes
et al., 2017). These are goal-oriented dialogues
in the domain of restaurant search. Here we tackle
Task 1, where in each dialogue the system asks the
user about their preferences for the properties of a
restaurant, and each dialogue results in an API call
which contains values of each slot obtained. Other
than the explicit API call notation, there are no an-
notations in the data whatsoever.

4.1 The bAbI+ dataset

While containing some lexical variation, the origin-
al bAbI dialogues significantly lack interac-
tional variation vital for natural real-life dialogue. In
order to obtain such variation while holding lex-
ical variation constant, we created the bAbI+
dataset by systematically transforming the bAbI
dialogues.

bAbI+ is an extension of the original bAbI Task
1 dialogues with everyday incremental dialogue
phenomena (hesitations, restarts, and corrections – see below). While the original bAbI tasks 2—7
increase the user’s goal complexity, modifications
introduced in bAbI+ can be thought of as orthog-
al to this: we instead increase the complexity of
surface forms of dialogue utterances, while keep-
ing every other aspect of the task fixed.

The variations introduced in bAbI+ are: 1. Hesita-
tions, e.g. as in “we will be um eight”; 2. Re-
starts, e.g. “can you make a restaur-

ant umm yeah can you make a restaurant
reservation for four people with french cuisine in
a moderate price range”; and 3. Corrections affect-
ting task-specific information - both short-distance
ones correcting one token, e.g. “with french oh no
spanish food”, and long-distance NP/PP-level
corrections, e.g. “with french food umm sorry
with spanish food”.

The phenomena above are mixed in probabilis-
tically from the fixed sets of templates to the origi-
nal data. The modifications affect a total of 11336
utterances in the 3998 dialogues. Around 21% of
user turns contain corrections, 40% hesitations,
and 5% restarts (they are not mutually exclusive,
so that an utterance can contain up to 3 modifi-
cations). Our modifications, with respect to cor-
rections in particular, are more conservative than
those observed in real-world data: Hough (2015)
reports that self-corrections appear in 20% of all
turns of natural conversations from the British Na-
tional Corpus, and in 40% of turns in the Map
Task, a corpus of human-human goal-oriented dia-
logues. Here’s part of an example dialogue in the
bAbI+ corpus:

sys: hello what can I help you with today?
usr: I’d like to book a umm yeah I’d like to book a
table in a expensive price range
sys: I’m on it. Any preference on a type of cuisine?
usr: with indian food no sorry with spanish food
please

4.2 Memory Network setup

In all the experiments we describe below, we fol-
low Bordes and Weston’s setup by using a
(we took an open source Tensorflow implementa-
tion for bAbI QA tasks and modified it according
to their setup – see details below). In or-
der to adapt the data for the , we trans-
form the dialogues into <story, question, answer>
triplets. The number of triplets for a single dia-
logue is equal to the number of the system’s turns,
and in each triplet, the answer is the current sys-
tem’s turn, the question is the user’s turn preced-
ing it, and the story is a list of all the previous turns
among both sides.

The hyperparameters are set as follows:
1 hop, and 128 as the size of embeddings; we train it for 100 epochs with a learning rate of 0.01 and
a batch size of 8 – in this we follow the best bAbI
Task 1 setup reported by (Bordes et al., 2017).

4.3 Testing the DS-TTR parser

Dynamic Syntax (DS) lexicons are learnable from
data (Eshghi et al., 2013a,b). But since the lexicon
was induced from a corpus of child-directed utter-
ances in this prior work, there were some construc-
tions as well as individual words that it did not
include. One of the authors therefore extended
this induced grammar manually to cover the bAbI
dataset, which, despite not being very diverse,

2See https://github.com/ishalyminov/babi_tools

3See https://github.com/ishalyminov/memn2n

4We are currently looking into applying Eshghi et al.’s
(2013a) model to induce DS grammars from larger seman-
tic corpora such as the Groningen Meaning Bank, leading to
much more wide-coverage lexicons
contains a wide range of complex grammatical constructions, such as long sequences of prepositional phrases, adjuncts, short answers to yes/no and wh-questions, appositions of NPs, causative verbs etc.

We parsed all dialogues in the bAbI train and test sets, as well as on the bAbI+ corpus word-by-word, including both user and system utterances, in context. The grammar parses 100% of the dialogues, i.e. it does not fail on any word in any of the dialogues. We assess the semantic accuracy of the parser on bAbI & bAbI+ using the dialogue-final api-calls in section 4.5 below.

### 4.4 Experiment 1: Generalisation from small data

We have now set out all we need to perform the first experiment. Our aim here is to assess the generalisation power that results from the grammar and our state encoding method (section 3.1) - we dub our overall model - and compare this to the state of the art results of Bordes et al. (2017). The method in Bordes et al. (2017) is not generative, rather it is based on retrieval of system responses, based on the history of the dialogue up to that point. Therefore, for direct comparison, and for simplicity of exposition, we do the same here: we apply the method described for creating a user simulation (section 3.2.1), this time for the system side, resulting in a 'system simulation'. We then use this to predict a system response, by parsing and encoding the containing test dialogue up to the point immediately prior to the system turn. This results in a triggering state, \( s_{\text{trig}} \), which is then used as the key to look up the system's response from the rules constructed as per section 3.2.1. The returned response is then parsed word-by-word as normal, and this same process continues for the rest of the dialogue. This method uses the full machinery of DS-TTR & our state-encoding method - the model - and will thus reflect the generalisation properties that we are interested in.

**Cross-Validation** Since we are here interested in data efficiency and generalisation we use all the bAbI and bAbI+ data - the train, dev, and test sets - as follows: we train Bordes & Weston's 2 and from 1-5 examples selected at random from the longest dialogues in bAbI – note bAbI+ data is never used for training in these experiments. This process is repeated across 10 folds. The models are then tested on sets of 1000 examples selected at random, in each fold. Both the training and test sets constructed in this way are kept constant in each fold across the & 2 models. The test sets are selected either exclusively from bAbI or exclusively from bAbI+.

### 4.4.1 Results: Predicting system turns

Table 1 shows per utterance accuracies for the & 2 models. Per utterance accuracy is the percentage of all system turns in the test dialogues that were correctly predicted. The table shows that can generalise to a remarkable 74% of bAbI and 65% of bAbI+ with only 5 input dialogues from bAbI. It also shows that 2 s can also generalise remarkably well. Although as discussed below, this result is misleading on its own as the 2 s are very poor at generating the final api-calls correctly on both the bAbI & bAbI+ data, and are thus making too many semantic mistakes.

### 4.5 Experiment 2: Semantic Accuracy

The results from Experiment 1 on their own can be misleading, as correct prediction of system responses does not in general tell us enough about how well the models are interpreting the dialogues, or whether they are doing this with a sufficient level of granularity. To assess this, in this second experiment, we measure the semantic accuracy of each model by looking exclusively at how accurately they predict the final api-calls in the bAbI & bAbI+ datasets. For the 2 model, we follow the same overall procedure as in the previous experiment: train on bAbI data, and test on bAbI+.

#### 4.5.1 Results: Prediction of api-calls

**BABBLE** Mere successful parsing of all the dialogues in the bAbI and bAbI+ datasets as shown above doesn’t mean that the semantic representations compiled for the dialogues were in fact correct. To measure the semantic accuracy of the DS-TTR parser we programatically checked that the correct slot values – those in the api-call annotations – were in fact present in the semantic representations produced by the parser for each dialogue (see Fig. 2 for example semantic representations). We further checked that there is no other incorrect slot value present in these representations. The results showed that the parser has 100% se-
memn2n on bAbI 2.77 59.15 70.94 71.68 ...

As we showed in section 4, even if most of these structures are not present in the training exam-

need to add or posit new machinery on bAbI 67.12 73.36 72.63 73.32 74.08
memn2n on bAbI 2.77 59.15 70.94 71.68 ...

mantic accuracy on both bAbI and bAbI+. This result is not surprising, given that DS-TTR is a general model of incremental language processing, including phenomena such as self-corrections and restarts (see Hough (2015) for details of the model).

MEMN2N Given just 1 to 5 training instances from bAbI as in the previous experiment, the mean api-call prediction accuracy of the 2 model is nearly 0 on both bAbI and bAbI+. This is not at all unexpected, since prediction of the api-calls is a generative process, unlike the prediction of system turns which can be done on a retrieval/look-up basis alone. For this, the model needs to observe the different word sequences that might determine each parameter (slot) value, and observe them with sufficient frequency and variation. This is unlike a semantic parser like DS-TTR, that produces semantic representations for the dialogues as a result of the structural, linguistic knowledge that it embodies.

Nevertheless, we were also interested in the general semantic robustness of the 2 model, to the transformations in bAbI+, i.e. how well does the 2 model interpret bAbI+ dialogues, when trained on the full bAbI dataset? Does it then learn to generalise to (process) the bAbI+ dialogues with sufficient semantic accuracy?

Table 2 shows that we can fully replicate the results reported in Bordes et al. (2017): the 2 model can predict the api-calls with 100% accuracy, when trained on the bAbI train-set and tested on the bAbI test-set. But when this same model is tested on bAbI+, the accuracy drops to a

<table>
<thead>
<tr>
<th># of training dialogues</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>on bAbI</td>
<td>67.12</td>
<td>73.36</td>
<td>72.63</td>
<td>73.32</td>
<td>74.08</td>
</tr>
<tr>
<td>2 on bAbI</td>
<td>2.77</td>
<td>59.15</td>
<td>70.94</td>
<td>71.68</td>
<td>72.6</td>
</tr>
<tr>
<td>on bAbI+</td>
<td>59.42</td>
<td>65.27</td>
<td>63.45</td>
<td>64.34</td>
<td>65.2</td>
</tr>
<tr>
<td>2 on bAbI+</td>
<td>0.22</td>
<td>56.75</td>
<td>68.65</td>
<td>71.84</td>
<td>73.2</td>
</tr>
</tbody>
</table>

Table 1: Mean per utterance accuracies (%) for datasets (10 folds)

very poor 28%, making any dialogue system built using this model unusable in the face of natural, spontaneous dialogue data. This is further discussed below.

5 Discussion

5.1

The method described above has the following advantages over previous approaches to dialogue system development:

– incremental (and thus more natural) language understanding, dialogue management, and generation;

– “end-to-end” method for task-based systems: no Dialogue Act annotations are required (i.e. reduced development time and effort);

– a complete dialogue system for a new task can be automatically induced, using only ‘raw’ data – i.e. successful dialogue transcripts;

– the MDP state and action spaces are automatically induced, rather than having to be designed by hand (as in prior work);

– wide-coverage, task-based dialogue systems can be built from much smaller amounts of data as shown in section 4.

This final point bears further examination. As an empirically adequate model of incremental language processing in dialogue, the DS-TTR grammar is required to capture interactional variants such as question-answer pairs, over- and under-answering, self- and other-corrections, clarification, split-utterances, and ellipsis more generally. As we showed in section 4, even if most of these structures are not present in the training exam-
ple(s), the resulting trained system is able to handle them, thus resulting in a very significant generalisation around the original data.

We also note that since we were in this instance interested in a direct comparison with 2s over the bAbI & bAbI+ datasets, we didn’t exploit the power of Reinforcement Learning and exploration as we described above - as we have done before with other systems (Kalatzis et al., 2016). Therefore the generalisation results we report above for follow entirely from the knowledge present within the grammar as a computational model of dialogue processing and contextual update, rather than this having been learned from data. Applying the full RL method described above would have meant that the system would actually discover many interactional and syntactic variations that are not present in bAbI, nor in bAbI+.

5.2 2

Even when trained on very few training instances, the 2 model was able to predict system responses remarkably well. But results from Experiment 2 above showed that this was misleading: the 2s were making a drastic number of semantic mistakes when interpreting the dialogues, both in the bAbI and bAbI+ datasets. Even when trained on the full bAbI data-set, the model performed badly on bAbI+ in terms of semantic accuracy. We diagnose these results as follows:

Problem complexity: The first thing to notice is that in bAbI dialogue Task 1, the responses are highly predictable and stay constant regardless of the actual task details (slot values) up to the point of the final api-calls; and further, that the prediction of api-calls is a generative process, unlike the prediction of the system turns, which is retrieval-based. This, in our view, explains the very large difference in 2 performance across the two prediction tasks.

Model robustness to the bAbI+ transformations: The variations introduced in bAbI+ are repetitions of both content and non-content words, as well additional incorrect slot values. The model was working in the same setup as , therefore none of those variations could be treated as unknown tokens for either system. Although in the case of 2, some of the mixed-in words never appeared in the training data, and bAbI+ utterances were augmented significantly with those words – so it was interesting to see how such untrained embeddings would affect the latent memory representations inside 2. The resulting performance suggests that there was no significant impact on 2 from these variations as far as the predicting system responses was concerned. But the incorrect slot values introduced in self-corrections affect the system’s task completion performance significantly, only appearing at the point of api-call predictions.

We note also that none of our experiments in this paper involved training 2 on bAbI+ data. There is a very interesting question here: is the 2 model in principle able to learn to process the bAbI+ structures if it is in fact trained on it? And how much bAbI+ data would it require to do so? These issues are address in detail in Shalymov et al. (2017).

6 Conclusions

Our main advances are in a) training end-to-end dialogue systems from small amounts of data, b) incremental processing for wider coverage of more natural everyday dialogues (e.g. containing self-repairs).

We compared our grammar-based approach to dialogue processing (DS-TTR) with a state-of-the-art, end-to-end response retrieval model (2s) (Bordes et al., 2017), when training on small amounts of dialogue data.

Our experiments show that our model can process 74% of the Facebook AI bAbI dataset even when trained on only 0.13% of the data (5 dialogues). It can in addition process 65% of bAbI+, a corpus we created by systematically adding incremental dialogue phenomena such as restarts and self-corrections to bAbI. We find on the other hand that the 2 model is not robust to the structures we introduced in bAbI+, even when trained on the full bAbI dataset.

Acknowledgements

This research is supported by the EPSRC, under grant number EP/M01553X/1 (BABBLE project6).

6https://sites.google.com/site/hwinteractionlab/babble
References


