Abstract

Data-driven dialogue systems are still far from understanding natural dialogue. Several aspects of natural language make it hard to capture in a system, such as unpredictability, mistakes and the width of the domain. In this thesis we take a step towards more natural data by examining disfluencies (i.e. mistakes). We test sequence to sequence models with attention on goal-oriented dialogue. Sequence to sequence models were chosen to overcome the unknown aspect of the mistakes, since they are known for their ability to generalise to unseen examples. The models are tested on disfluent dialogue data, the bAbI+ task, in addition to normal goal-oriented dialogue data, the bAbI task. In contrast to previous findings with memory networks, we find that the sequence to sequence model performs both the bAbI tasks as the bAbI+ task well achieving near perfect scores on both tasks. A slight decrease in performance is noticed when introducing disfluencies only to test data, only 80% accuracy is measured in this condition. This is surprising because memory networks are very similar to sequence to sequence models with attention.

The results of the main experiment suggest that sequence to sequence models learn to parse disfluencies. Attention visualisation results suggest that the bAbI+ model does indeed learn to pay attention to disfluencies in a meaningful way. Even though attention shows that the model is aware of disfluencies, further analyses using diagnostic classifiers and diverse inputs suggest that the encoder is not learning to parse disfluencies, as we expected, but functions more as a memory. The decoder in turn appears to access the encoder as this memory using the attention mechanism, which proved crucial to learning the bAbI tasks.

Acknowledgements

I thank Dieuwke Hupkes for her supervision, time and guidance. She was a great sparring partner to bounce ideas of.

I thank Raquel Fernandez for her supervision, time, guidance. Without her expertise this thesis would not have been the same.

Lastly I thank Sander Meester, Dieuwke Hupkes and other people for proofreading.
# Contents

1 Introduction ................................................. 4  
   Contributions ............................................. 5  

2 Background .................................................. 6  
   1 Dialogue Systems ........................................ 6  
      1.1 End-to-end dialogue systems ....................... 8  
      1.2 Characteristics of Data ............................ 10  
   2 Disfluencies .............................................. 11  
   3 Interpreting Networks .................................... 14  

3 Experiments .................................................. 18  
   1 Data ..................................................... 18  
      1.1 bAbI .................................................. 20  
      1.2 bAbI+ ............................................... 21  
   2 Methods ................................................... 23  
      2.1 Models ............................................... 23  
      2.2 Training ............................................. 23  
      2.3 Experimental Setup .................................. 24  
   3 Results ................................................... 25  
      3.1 Parameter Search: Hidden Layer Size ................ 25  
      3.2 bAbI Tasks ......................................... 26  
      3.3 bAbI+ Task ......................................... 27  

4 Analysis ....................................................... 29  
   1 Generalising to Disfluencies ............................ 29  
   2 Attention ............................................... 32  
      2.1 Visualisation ....................................... 32  
      2.2 No Attention ....................................... 36  
   3 Editing Term ............................................. 37  
      3.1 No Editing Term ..................................... 38  
      3.2 Generalisation over Length of Editing Term ....... 39  
   4 Incrementality ............................................ 41  
      4.1 Triggering API-calls ................................ 41  
      4.2 Diagnostic Classifiers ............................... 43  
      4.3 Progression Visualisation ........................... 44  
   5 Disfluency Theory Diagnostic Classifiers ............... 45  

5 Discussion .................................................... 49  

List of Figures ................................................. 52  
List of Tables .................................................. 53  
Bibliography .................................................... 57
CHAPTER 1

Introduction

Dialogue systems are widely used, ranging from personal assistants and chat bots to very specific services that allow you to make a reservation or complaint. For example, one can book a restaurant through dialogue with a system, learn about their appointments, set reminders, and more by interacting with dialogue-based bots. These dialogue systems all tackle the challenge of conducting dialogue. This is hard due to several characteristics of natural language, which are often unseen in the data. Human language usage can be unpredictable and full of inconsistencies, certainly when mistakes are made.

In this thesis, we extend research that uses a sequence to sequence model as a dialogue system. Models were implemented with two LSTM’s (long short term memory network) in an encoder-decoder set up with attention. With this dialogue model, we try to answer whether sequence to sequence models can learn task-oriented dialogue and whether sequence to sequence models can generalise to disfluencies in this data.

One can interrupt dialogue by hesitating and saying "uhm" or by making and correcting mistakes. This interruption is called a disfluency. In this thesis, we focus on hesitations and corrections, mentioned in the previous sentence, and restarts. Restarts are similar to corrections, but instead of repairing part of the sentence, the speaker starts anew. Examples of a hesitation, a correction, and a restart are shown in Example 1 in that order. The words that constitute the interruption are underlined and indicated in red.

Example 1. I would like uhm tea please.
I would like tea no coffee please
I would like tea... Can I get a coffee?

Models have struggled with these inconsistencies in the data and often this is resolved by filtering out disfluencies and other irregularities with regular expressions and similar methods. More recent research shows that disfluencies actually carry information [10]. An example of this is Example 2, shown below. In this example, filtering the sentence results in "He quit his job" which does not tell you who lost his job. This information is lost when the disfluencies are filtered out.

Example 2. "Peter got f... he quit his job"

In contrast to filtering disfluencies out, a dialogue system can directly parse disfluencies. This allows the model to keep all information in memory which can be useful for processing dialogue data. Additionally, this makes the approach of the dialogue system more similar to a human approach to dialogue processing. There is substantial evidence that humans do remember corrected interruptions, even after processing the disfluency.

Using sequence to sequence models for a dialogue system is informative for cognitively plausible dialogue systems. In other words, dialogue systems that process dialogue as a human would. There are several aspects of sequence to sequence models that make them similar to humans. The first is incrementality, when each word is processed and integrated with everything up until that point in contrast to collecting the entire sentence or utterance first and then processing it as
a whole. Sequence to sequence models are by nature incremental, which makes them more suited for cognitively plausible dialogue systems. The second is end-to-end processing: an end-to-end model is trained from input to output directly, in contrast to training all steps and parts of the model separately. This is similar to how humans learn from input and output only. The last aspect of human intelligence that is represented well in sequence to sequence models is generalisability. Humans are adaptive in their thinking and acting, and can understand and create sentences that they have never seen or heard before. Similarly, sequence to sequence models can generalise well to unseen phenomena. An example of this is from computer vision where zero-shot learning based on linguistic description has shown promising results (for example [39] and [24]). This generalisation ability, besides making the model more cognitively plausible, may help the model with generalising to disfluencies that are introduced in the data.

1.1 Contributions

End-to-end dialogue systems, such as the dialogue systems tested in this thesis, have been shown to perform well on other tasks [44]. However, they have not been widely tested on small domains. In this thesis we test our sequence to sequence dialogue systems on goal-oriented dialogue within the restaurant booking domain.

Studying dialogue systems that are “cognitively plausible”, in other words systems that act like a human might, is a field of study that attempts to unravel human intelligence by reproducing it artificially. On the one hand we contribute to this by using a sequence to sequence model which is similar to humans as explained above. On the other hand, analysing the model both on aspects that are and aspects that are not similar to humans is a starting point for improving towards this goal.

Importantly, we contribute to the field of diagnosing neural models, which is still small and has much unexplored territory. In this thesis we use diverse and complementary analysing techniques, exploring a wide range of possibilities. Uniquely this thesis uses psycholinguistic theory to base hypotheses off, which are combined with the different analysis methods to really get to the core of the model’s workings.

We compare our sequence to sequence dialogue systems to an approach using memory networks, further establishing the similarities and differences between memory networks and sequence to sequence systems. Other well-established systems such as rule based systems are more commonly used. In this thesis we thus contribute to diversity in dialogue systems and we contribute to research with neural models as dialogue systems.

Little is known about how these deep learning models do generalisation and whether or not they perform this generalisation in any way similar to how humans do it. In this thesis we dive into the generalisation abilities of these systems through several analyses. This teaches us several things about the working of the deep learning models and when the model can or cannot generalise.

1.2 Overview

In the next chapter, we go over the required background knowledge for the experiments and models. We first revisit dialogue systems, their history and the working of relevant models. Next we discuss background knowledge on disfluencies and lastly we discuss different methods for analysis of neural models.

In the third chapter we reproduce both the tasks in the bAbI paper [3] as the experiment done in the paper of Shalyninov et al. [38] with sequence to sequence models. We go over the design choices for our sequence to sequence dialogue system, the experiments performed to evaluate and validate them, and the results of these experiments.

In the fourth chapter we diagnose the models described in the third chapter and go into some aspects of disfluency theory in regard to our model’s performance.

In the fifth chapter we conclude the thesis with a discussion and a conclusion.
CHAPTER 2

Background

Before diving into the dialogue systems build in this thesis, we go over the relevant background knowledge.

First we introduce a general history of dialogue systems: We discuss different dialogue systems and how the design of them evolved over time. At the end of this time-line we branch off into end-to-end dialogue systems. Here we introduce general characteristics of end-to-end dialogue systems, which are often neural like the dialogue system implemented for this thesis. Which is what we go over next: The mechanisms used in the design of the dialogue systems of this thesis. We end this section by touching upon an important part of end-to-end dialogue systems: the data used to train the systems. We discuss important characteristics of this data and the advantages and disadvantages of each.

Later in this thesis we introduce the experiments that these thesis are based upon. First, we try to determine whether sequence to sequence models can learn goal-oriented dialogue. Second we test the generalisability abilities of said sequence to sequence model to disfluent data. As a background to the second experiment we go into theory on disfluencies. In the second section of this chapter we start with the general form, distinguishing types and considering some distributional properties. After establishing some understanding of disfluencies, we go over relevant literature on handling and parsing of disfluencies. Here we discuss important milestones from both psycholinguistics and computational linguistics.

Finally, we conclude the background chapter with a section on understanding and diagnosing of neural deep learning models in general. The models trained as dialogue systems in the experiments that are central to this thesis are neural models. There is a lot of research into analysing and understanding these models and opening the “black box”. In this final section we discuss a wide range of analysing methods, ranging from traditional to unknown and from superficial to deep into the network.

1 Dialogue Systems

Dialogue serves many purposes: It is used for communication, sharing information, coordinating actions, and coordinating the dialogue itself (e.g. turn-taking). To train dialogue systems on this diverse task, interactive spoken dialogue is often used as examples to train dialogue systems with. Training with spoken language introduces many challenges by itself, even though advances in deep learning have overcome many of them. Where written language adheres to structure such as sentences and is usually grammatically coherent, speech is often less structured. Dialogue data itself introduces some challenges as well. One example are the concept “utterances”: a semi-coherent set of words uttered by one speaker. If one speaker finishes the sentence of another, there are two distinct utterances, one per speaker. However understanding of the utterance cannot be performed as a standalone task as easily as with traditional sentences. In dialogue, all utterances are exchanged not simultaneously but in turns, where a turn is defined as all utterances one speaker does within a conversation without anyone else uttering in between.
Utterance segmentation and speech recognition are at the basis of any system that conducts dialogue. Interpretation of the utterances is performed by different dialogue systems in different ways, in this section we go over different approaches to this, with which we place our own approach in context.

Dialogue systems are used in a wide range of applications, such as technical support services, digital personal assistants, chat bots, and home entertainment systems. Despite the wide range of applications, the success of dialogue systems, specifically in unbounded domains, is still lacking. The most successful dialogue systems are used in a narrow domain where a clear goal is to be achieved through dialogue, called goal-oriented dialogue. In this setting a system needs to understand the user request and the information needed to perform it and complete the related task or goal within a limited number of turns. This thesis focuses on neural dialogue systems trained on a goal-oriented task. Alternative approaches are laid out first, after which the section dives into explaining the end-to-end neural dialogue system.

### Rule-based approaches

The early successes of dialogue systems were achieved using hand-crafted rules and grammars. These systems are by nature hard to scale and re-use [3].

Another early strategy for dialogue systems was centring the modelling around pre-defined slots. One of the first instances of this was the “Information State Update” dialogue system introduced by Lemon et al. [25]. They introduced a dialogue strategy based approach to dialogue systems where different slots of a state are filled with information. The dialogue system was trained through reinforcement learning. Another approach was introduced by Wang and Lemon [45], where beliefs are maintained over slots using domain-independent rules and basic probability operations. This approach results in a general dialogue state tracker which does not require any training, which is more efficient when applied to a known domain. Even though such “slot-filling” approaches have proven successful they are hard to scale to new domains and less restricted dialogue.

### Data-driven approaches

In contrast to rule-based approaches there are machine learning approaches, which have shaped the design of more recent dialogue systems. Speech recognition software has greatly increased performance due to innovations in deep learning [2], [14]. This introduced new challenges: where user satisfaction dialogue length and goal completion rates were sufficient in dialogue before, automatic optimisation criteria are required when training machine learning systems.

Data driven machine learning has proven effective for a wide range of dialogue related NLP tasks. Stolcke et al. introduced a dialogue system that models dialogue with Markov models as a likely sequence of dialogue acts [41]. With this approach they improved on the then state-of-the-art of both speech recognition and dialogue act classification.

Another approach based on Markov models is introduced by Young et al. who use an explicit Bayesian model of uncertainty optimised with POMDPS(Markov Processes) for dialogue policy learning [49]. Modelling the uncertainty allowed them to bypass the errors introduced by speech recognition errors and similar confounding factors. Dialogue policy learning is very similar to the slot-filling mechanism explained above.

The approach based on beliefs, also called dialogue state tracking was also attempted with machine learning approaches. Henderson et al. [12] originally introduced the dialog state tracking challenge to compare multiple approaches to dialogue state tracking or belief tracking on a shared task. Additionally, the paper introduced their own deep neural network approach to dialogue state tracking, which performed adequately on the task. Henderson et al. [13] later introduced another deep learning based model which directly maps the output of the speech recogniser to a dialogue state without an explicit semantic decoder. Their model set the new record for the dialogue state tracking approach.

Another field in which machine learning approaches are taking the lead is natural language generation. Langkilde and Knight [23] started with statistical approaches, after which stochastic learning took flight [31], [33]. Attempts were made with graphical models by Mairesse et al. [30] and even an LSTM (long short term memory) approach by Wen et al. [46].
1.1 End-to-end dialogue systems

An interesting development in data-driven dialogue modelling is end-to-end training, in other words training from input to output without any supervision or sub-components. This way of training has proven to be very promising ([44], [40], [37]). Input is usually text or speech and output is either a textual response or a distribution over possible responses. All model parameters are optimised with respect to a single objective function, often the maximum of the log-likelihood on a fixed corpus of dialogues, in general really big. In their purest form the models only depend on dialogue context, but they can be extended with outputs from other components or with external knowledge, such as the knowledge base provided in the bAbI tasks.

Some models deterministically select from possible utterances, such as the models trained as baselines for bAbI and bAbI+. All information retrieval based systems and systems based on reranking systems are of this type. Some neural models are of this type, such as the model created by Lowe et al. [29], which computes a relevance score over a set of candidate responses using a recurrent neural network.

The models in this paper however are of a different type: they are so-called generative models which generate utterances word by word by sampling from a full posterior distribution over the vocabulary (for example [44] and [35]). Advantages are that models are able to generate completely novel responses. However, generating is by nature more complex than deciding and thus re-ranking based methods that do not use any generating, making generative models more expensive computationally. When trained solely on text these models are a form of unsupervised language learning, since they are learning the probabilities of every possible conversation, making them truly end-to-end.

Sequence to sequence models

In this paper the data-driven goal-oriented dialogue system is of the type generative neural model, in particular a sequence to sequence model. The sequences in this are the input sequence (the dialogue history + the current human utterance) and the output sequence (the bot utterance). Sequence to sequence modelling allows certain freedoms, such as variable input and output lengths, which allow it to generalise well over a range of examples.

The encoder-decoder design [4] provides a pattern for using recurrent neural networks in a sequence to sequence fashion. They consist of two models, in our case LSTM’s (long short term memory) [15]: an Encoder and a Decoder. Intuitively, the first LSTM encodes the information in the input into a hidden state, a vector representation. This hidden state is then fed into the second LSTM, the decoder, which decodes this hidden state into the output sequence. The encoder-decoder approach has proven to work well even on very long sequences [43].

Long Short Term Memory

The encoder-decoder framework we use consists of two LSTM’s. An LSTM (long short term memory) is a unit for sequential processing and was first introduced by Hochreiter and Schmidhuber [15]. An LSTM is often used as a building block of a recurrent network structure, the entire structure is then referred to as an LSTM model. An LSTM block consists of a cell state, an output/hidden state, an input gate, an output gate and a forget gate. Since the hidden state has limited size the model must prioritise, which the model does through the different gates in an LSTM. At each step the weights of the framework determine which information to keep, which information to discard, and how to store this information through computations in the gates, often using the logistic function. The input gate determines which part of the input to discard and which part of the input will flow into the hidden state. The forget gate determines what in the cell state to forget. The output gate determines what part of the cell state will determine the output.
Figure 2.1: LSTM layout, source: stratio

Where $f_t$ is the forget gate, $i_t$ is the input gate, $o_t$ is the output gate, $x_t$ is the input vector, $c_t$ is the cell state and $h_t$ is the hidden state. The hidden state is the same as the output state.

$W$ and $U$ are both weight matrices and $b$ is the bias vector, all of which are learned during training.

**Attention**

A limitation of the sequential processing in the encoder decoder in combination with the fixed length hidden state in an LSTM is that input vanishes over time, which introduces problems for very long sequences. An interesting mechanism that sequence to sequence models can use to overcome this is attention. The mechanism is loosely based on human visual attention. It allows the model to prioritise certain parts of the input directly, even words at the beginning of the sentence. Attention was introduced by Bahdanau [1].

The attention mechanism is implemented on the encoder decoder design. The model is trained to learn which states to pay attention to. Each intermediate output, the hidden state at each sequential step, of the encoder is saved. Each decoder output word now depends on a weighted combination of all these intermediate output states, instead of just the output state after the final word the encoder encounters. It thus comes down to the computation of a vector over the inputs as input to the decoder.

$$a_{ts} = \frac{\exp(score(\vec{h}_t, \vec{h}_s))}{\sum_{s'=1}^S \exp(score(\vec{h}_t, \vec{h}_{s'}))} \quad \text{(2.6)}$$

$$c_t = \sum_s a_{ts} \ast \vec{h}_s \quad \text{(2.7)}$$

where $c_t$ is a context vector, the vector that encompasses all output vector of the encoder (all $h_s$’s) and $a_{ts}$ are the attention weights. The contributions of the inputs sum to 1.

In addition to the accuracy benefits, attentions allows us to inspect and visualise what the model is paying attention to directly.
Saving these states and computing their contribution increases the computational burden of the models. The computations behind attentions are notably similar to the output memory representation in memory networks, which are explained in more detail below.

Memory networks

Bordes et al. [3] presented memory network results for the bAbI task in their paper, suggesting it as a final baseline for neural models on the bAbI task. Since we will be comparing our model to this baseline, it is important to know to what extent our models are comparable to memory networks.

Memory networks were introduced by Weston Chopra and Bordes [47]. As the name suggests, this approach includes a long-term memory component which can be read and written to directly, often serving as a sort of knowledge base. Memory networks reason with inference components combined with a long-term memory component with the goal of prediction.

Memory networks process in four steps: First, it converts $x$ to an internal feature representation. Second, it updates memories $m_i$ given the new input. Then, it computes output features $o$ given the new input and the memory. Finally, it decodes output features $o$ to give the final response.

$$m_i = G(m_i, I(x), m), \forall i \quad (2.9)$$
$$o = O(I(x), m) \quad (2.10)$$
$$r = R(o) \quad (2.11)$$

where $x$ is the input; $I(x)$ is the internal feature representation; $G$ is the function that updates old memories based on new input; $O$ produces a new output; and $R$ converts that output into the desired format.

Memory networks were shown to outperform both LSTM’s and RNN’s (Recurrent Neural Networks) on question answering tasks [47]. Similar to goal-oriented dialogue, question answering requires a model to grasp the red lining of a conversation or story.

Sukhbaatar et al. [42] tested end-to-end trained memory networks on question answering and language modelling, showing that testing end-to-end improves results on end-to-end but more importantly that training end-to-end makes results for language modelling comparable to LSTM’s and RNN’s. Their results indicate that multiple computational hops (=Multi-step reasoning) do indeed yield improved results.

Perez et al. [34] introduced an architecture based on and similar to memory networks, a memory access regulation mechanism. They based it on the connection short-cutting principle which proved successful in computer vision.

An important aspect of all these memory networks is the inference included in the model. The inference can consist of multiple steps, called multi-step or multi-hop. A normal sequence to sequence model with attention can be seen as a memory network with one hop, since the attention allows it to access all remembered facts when “inferring” the output. Inference is not an explicit component of encoder decoder models, however, there are still many similarities between the two model types that make the comparison meaningful.

1.2 Characteristics of Data

For data-driven machine learning systems the data that is used to train and to use the system defines the quality of the system obtained. A marked example of this is Microsoft’s AI-based chat-bot named Tay, who was terminated after it became racist due to bad input from users. Common practices and defining characteristics of data are discussed below.

The first characteristic to take into account is the origin of the data: human-human and human-machine. Human-human data, such as the switchboard corpus, are a great source for more natural language usage, which often is considered the end-goal of dialogue research. Human-human dialogue features more involved turn-taking, more pauses and more grounding problems.
However, the domain is unbound and the scope and variance of linguistic phenomena is not defined or controlled, resulting in unbalanced training data.

On the other hand human-machine data excels at defining a scope or a domain. Linguistic features in target data (machine utterances) can easily be contained, counted and tweaked. More synthetic approaches to data acquisition may even control linguistic features and other variances in human data as well. This comes at a cost as the diversity and variability of the machine utterances greatly decrease, as stated by Williams and Young [48].

There is a distinction between goal-oriented dialogue and general dialogue, of which we focus on the prior. Goal-oriented dialogue is dialogue within a narrow domain such as restaurants, where there is a distinct goal to be achieved through dialogue such as ordering food. The dialogues are clearly used as a means to an end, and can more easily be graded since they either achieve said goal or don’t. In contrast to this is general dialogue, unbounded dialogue, or unconstrained dialogue, which has no rules and serves more purposes than a single goal. This is a characteristic we know from most human dialogues. Goal-oriented dialogue modelling was originally based on deterministic hand-crafted rules. In 1990s research into machine learning for classifying intent and bridging the gap between text and speech took off, when markov decision processes were used more. Research in general dialogue has progressed, however commercial system are still highly domain-specific and based on hand-crafted rules and features. The dialogue system in this thesis is also domain-specific, but is not based on rules and features and is trained completely end-to-end.

2 Disfluencies

Natural language is a domain that is non-trivial to model. Many things contribute to this complexity of language, among them the mistakes and inconsistencies introduced by human speakers. Many forms of mistakes and inconsistencies exist and the occurrence and handling of these phenomena has been studied by psycholinguists for a long time. Another group of researchers interested in studying disfluencies are computational linguists, their research is closer to what is presented in this thesis. Important findings from both directions are laid out in this section, which is the basis for linguistically informed analysis performed on the models in this thesis.

General form and subcategories

Levelt [26] unravelled an underlying structure to disfluencies. His work show-cased clear rules both listeners and speakers adhere to when introducing and parsing disfluencies.

Following the structure Levelt unravelled, Ginzburg et al. [10] divided a disfluency into five parts: the start, the reparandum, the editing term, the alteration, and the continuation. Table 2.1 shows an example of a correction from the bAbI+ dataset, divided into this structure.

<table>
<thead>
<tr>
<th>start</th>
<th>reparandum</th>
<th>moment of interruption</th>
<th>editing term</th>
<th>alteration</th>
<th>continuation</th>
</tr>
</thead>
<tbody>
<tr>
<td>book</td>
<td>a french</td>
<td></td>
<td>uhm sorry uhm</td>
<td>a vietnamese</td>
<td>restaurant</td>
</tr>
</tbody>
</table>

In this example the editing term is “uhm sorry uhm”. The starting sentence and the continuation are the correct sentence on either side of the interruption, in our example “book” and “restaurant” respectively. The reparandum is the “wrong” sequence to be repaired and the alteration is the sequence to be inserted in its place, in our example “a french” and “a vietnamese” respectively. In the middle of the reparandum and the editing term is the moment of interruption, which is implicit and has to be detected by the listener and an editing term. All but the continuation and the moment of interruption are optional.

Which of these five parts are present can be used as a basis for dividing disfluencies into subcategories. One distinction to make is how the alteration relates to the reparandum. Three big distinctions in this respect which all have a reparandum and an alteration are: A repair
(called correction in the bAbI+ dataset), where the alteration replaces the reparandum, as seen in example 3; A reformulation, where the alteration elaborates on the reparandum, as seen in example 4; and a false start, where the alteration differs strongly from the reparandum, as seen in example 5.

**Example 3.** I would like a french uhm sorry uhm a vietnamese restaurant

**Example 4.** I would like a french uhm sorry uhm a parisian restaurant

**Example 5.** I would like a french uhm sorry uhm. Let’s look at our schedules for tomorrow.

Of these three types of disfluencies, repairs(corrections) are still a broad and diverse term. Further sub-classification exists based on the reason for the repair. A repair where the alteration replaces content that is inappropriate to express the message or inappropriate for the audience is called “appropriateness-repair”. An example of such a repair can be seen in example 6, here the speaker found “there” too vague and thus not appropriate. A repair to replace erroneous content is called a “error-repair”. The repair shown before in example 3 is an error-repair.

**Example 6.** ”from there, from the blue node” [26]

One type of disfluencies, dubbed forward looking disfluencies by Ginzburg et al. [10], differs from the previously discussed disfluencies in the absence of a reparandum. In other words, there is no error to be repaired. These forward-looking disfluencies are also divided into subcategories, with three main subtypes: Hesitations, where there is no alteration and continuation is simply delayed by the interruption; restart, where the alteration is the same as the reparandum; and restarts, where the alteration is the same as the start+reparandum. Examples of a hesitation and a restart are shown below.

**Example 7.** we will be uhm eight

**Example 8.** good morning uhm yeah good morning

**Parsing disfluencies**

Now that we know the general structure of disfluencies we can look into theory about parsing disfluencies. The following subsection will discuss psycholinguistic and computational linguistic approaches to this.

Levelt introduces several theories and interesting insights into the introducing and parsing of disfluencies as done by humans. He introduces important characteristics that speakers adhere to. Additionally he discusses multiple aspects of what a listener is theorised to do to process disfluencies. All these aspects are related back to the five parts of a disfluency as explained in Table 2.1.

One important problem when detecting disfluencies is that the moment of interruption is not directly measurable. The editing term is used to notice the interruption when it is present. This is often not the case, only in 18.52% of corrections an explicit editing term is present in the Switchboard corpus [16]. Other properties of the disfluency are used for detecting the interruption when there is no editing term.

After the detection of an interruption the listener is faced with two tasks. The first task is recognizing what type of disfluency he is dealing with, which is vital for interpretation of the disfluency. The second task is identifying what the reparandum is and what the alteration is, if present. Depending on the outcomes, the alteration is then merged with the original utterance(OU) or interpreted as a restart. Figuring out what is the reparandum is dubbed the continuation problem by Levelt [26].

Listener’s are expected to pay attention to two aspects when solving this problem. The first aspect is the syntactic category of the first word of the alteration denoted with $r_1$. If $r_1$ is the same as $o_n$ the last word of the reparandum the sentence is continued from $o_n$ after being replaced by $r_1$. An example of how humans use these disfluencies from Levelt’s research [26] can be seen in example 9.

**Example 9.** Van het groene rondje naar boven naar een roze . . ., oranje rondje

*From the green disc to up to a pink . . ., orange disc*
The second aspect is the lexical identity of $r_1$. A strategy suggested would be to check if there is a word in OU, let’s say $o_i$ that is identical to $r_1$ then replace $o_i$ with $r_1$ and insert the alteration from there on. An example of this is shown in Example 10, where *naar*(to) is $r_1$ and $o_i$. Example 3 introduced earlier in this chapter follows this behaviour, here the similarity is a in a *vietnamese* and a *french*.

**Example 10.** Rechtsaf naar geel, eh naar wit
Right to yellow, uh to white

**Example 11.** with sorry yeah with british cuisine

Different strategies apply to other disfluency types, such as the forward looking disfluencies introduced by Ginzberg et al. [10] Examples of forward looking disfluencies are hesitations (Example 7) and restarts (Example 8). For a hesitation a listener is theorised to continue processing as if no interruption had taken place, after determining whether the continuation is easily integrated with the start of the sentence. If the continuation does not integrate with the start of the sentence, i.e. in case of a restart, this is more complex. For restarts, when the disfluency is classified as such, the listener is expected to disregard everything prior to the moment of interruption and replace the entire beginning of the sentence with the fresh start.

One thing that is consistent across different types of disfluencies is that which disfluency and how to interpret the information seems to be decided always on $r_1$.

A model that adheres to many of these principles is the Tree Adjoining Grammar, introduced by Joshi et al. [19]. The method is centred around parsing and triggers when the incremental parser happens upon new material that cannot be attached to an existing node. In our example this is the case when two different slot values are present in one sentence. The method attempt to overlay the previous tree with the new interpretation. It recognises where to attach the root, by detecting similarities as suggested by Levelt [26]. The new tree does replace the old tree, but it is not completely deleted and can still influence further processing. Recreating processing effects such as lingering, which refers to memory of mentions fading over time instead of instantly.

**Importance of disfluencies**

Disfluencies are traditionally considered separate from language and thus dialogue abilities. This stems from the competence versus performance view introduced by Chomsky [5] where disfluencies are denoted as accidental mistakes that are not part of a speaker’s competence. Speech, including disfluencies, is dubbed performance, in contrast to the inherent knowledge and ability of a speaker which is dubbed, as mentioned, competence.

In line with this dialogue systems were originally not taught to process disfluencies, instead disfluencies were filtered out of the data leaving only the “intended” sentence. One way of doing this is by using rules and grammar-like structures, as for example McKelvie did [32]. Another way of doing this is by leveraging statistical information, as done by Heeman and Allen [11]. These studies come from computational linguistics. Studies in this field give many insights into the structural properties of and occurrence of particular forms and or types (distributional characteristics). These characteristics have been used to automatically detect and handle disfluencies in contrast to mimicking humans who parse and sometimes remember the reparandum of a disfluency.

Detection of disfluencies served the purpose of filtering them out during parsing, before any semantic interpretation. Filtering out disfluencies is practical for transforming dialogue data and applying dialogue systems to it. However, filtering out disfluencies is not a plausible model for human handling and interpretation of disfluencies, as argued by Ferreira, Lau and Bailey [8].

Schegloff, Jefferson, and Sack [36] did a study outside the previously explained paradigm, and looked into the similarity between self-corrections and clarification requests. Where clarification requests are a certain type of question that can be asked in dialogue and which is thus included within dialogue research. They related the structure of this type of disfluencies which were not considered language to clarification requests. Thus giving rise to the line of work that considers disfluencies an integral part of language and thus a subject worthy of study.
Ginzburg, Fernandez and Schlangen [10] argue for disfluencies being a grammatical phenomenon. They stated that disfluencies have a significant cross linguistic variation, which is an important characteristic of grammatical phenomena.

More important than the grammatical phenomena identity of disfluencies, is whether or not they contribute to conduction of dialogue. After the occurrence of a disfluency, the continuation can refer back to the reparandum. An example of a disfluency with such a reference is shown in Example 12.

**Example 12.** *I heard that Peter got fired uhm that he quit his job.* [10]

When simply detecting and fixing this disfluency the resulting sentence is incomplete and misses crucial information. For the example above the resulting sentence is:

**Example 13.** *I heard that he quit his job.* [10]

Another argument for considering disfluencies as a part of language, and thus arguing for equipping dialogue systems with the ability to handle them is found in lingering. Lingering is the notion of previously mentioned words that were repaired “lingering” in understanding or parsing mechanisms, thus influencing further processing. Human interlocutors are known to remember the content of the reparandum even when the correction is used for further reasoning. Feirrrara et al. [8] did a study where human participants evaluate the grammaticality of sentences with and without disfluencies. Interesting in their results is that sentences that were ungrammatical with the correction in place but grammatical when considered with the reparandum, were judged to be grammatically more correct than sentences without such a reparandum, proving that humans do remember the reparandum. For example people find example 14 more grammatical than example 15.

**Example 14.** “*Mary will throw uhm put the ball”* [8]

**Example 15.** “*Mary will put the ball”* [8].

Clark, Tree and Fox [6] claimed that filled pauses (hesitations) are lexical items with the conventionalised meaning “a short/slightly longer break in fluency is coming up”. They propose that disfluencies are genuine communicative acts of speakers used to achieve synchronisation. In other words, speakers use disfluencies to communicate about breaks in fluency, other dialogue coordination and general dialogue aspects.

These lines of research show that there is information in disfluencies which is lost when simply detecting and parsing disfluencies into “good” sentences. Understanding the role of disfluencies in language understanding and dialogue still leaves room for research.

Neural models are known for their ability to pick up on regularities in the data. Relevant information in the reparandum could very well be picked up by the model trained in this research. To determine whether they picked up on these aspects from theory we analyse the models. Different techniques for analysing neural models are described in the next section.

## 3 Interpreting Networks

Deep neural networks are unintuitive to interpret, due to the high dimensionality and highly interconnected layers in them. Both understanding how the models are performing their tasks, what is influencing the results, mistakes in the data or other confounding variables are hard to distinguish from mistakes in complex neural architectures. Many functionalities of models are part of the so-called black box. Several attempts have been done to break open this black box, an overview of research on interpreting models similar to the methods in this thesis are discussed below.

**Traditional analyses**

The most accessible and intuitive high-level analysis is accuracy. Accuracy can tell you a lot about the functioning of a model in respect to the task. However it gives very little insight into
what the model is doing. It is difficult to translate theory into predictions for accuracy and testing them, without the influence of confounding variables.

Accuracy is thus often lacking for explaining what a model is doing, an intuitive next step is the inspection of output of the model, in other words error analysis. Error analysis gives many insights into the workings of the model, looking at some -possibly cherry picked- examples and hypothesising about the models workings is qualitative error-analysis. In addition to qualitative error-analysis, a researcher could do quantitative error-analysis such as classifying errors into different categories. How successful error analysis is depends on the complexity of the data and the model inspected. For more complex models, it is most useful as a starting place for hypotheses of what the model is doing, and what can be improved on.

**Tweaking input**

When error analysis does not give enough insights, because the data is too complex or the results are very ambiguous, a method of more precise inspection is tweaking the input. This is a method of analysing the model through carefully designed input-output pairs, tweaking input such that it does or doesn’t have certain grammatical characteristics. This method is similar to error analysis, but is far more specified towards, though not limited to, linguistic phenomena.

Linzen, Dupoux and Goldberg [28] showed this type of inspection for subject-verb dependencies. They probed the architecture’s grammatical competence by designing input where the model either has to predict the number of the subject when it would output the verb. The input the model was tested on includes examples with agreement attractors and distractors. Additionally examples were made with relative clauses and without, to see if the model was able to understand the more complex structure of a relative clause.

The bAbI+ dataset is already strategically designed and is thus vaguely related to this approach of network inspection. Code for data generation is openly available, allowing any researcher to tweak the percentages of disfluencies and alter the patterns used for generating them.

There are other regularities in the bAbI data, such as sentences that often elicit certain responses. Another way of tweaking the input is exploiting this and augmenting the input with these “probing” sentences. This allows inspection of normally unseen parts of the model, for example in the bAbI dataset one can view the API-call mid conversation, while using the native mechanisms. This can help identifying whether problems are in the data or in the model by forcing the model to predict things it would not normally predict. Among other things, this gives insights into the order of processing.

These methods fail to inspect layers of the models that are far from input and not otherwise directly related to input and output. The mechanisms that cannot be inspected through tweaking of input are often less intuitive in interpretation altogether, asking for more specialised diagnosing and analysing methods, which are described below.

**Visualisations**

Visualisation of values within the architectures is a method that angles inspection towards deeper layers or mechanisms of the model. There are many numbers in machine learning models, values of different layers and mechanisms such as attention or gates for an LSTM. Visualisations can be used to bridge the gap between these values which are only understood by the machine, to pictures that humans can interpret far more easily.

Karpathy et al. [21] used visualisation to get a better understanding of the internal dynamics of a character-based recurrent neural network. They plotted individual cell activations which allowed them to find several cells that performed interpretable behaviour, such as keeping track of the scope of quotes or representing the length of a sentence.

Instead of character-level models, visualisations have also been used to interpret word-level recurrent neural nets. Li et al. [27] inspected a word-level model, they focused on visualising compositionality in sentiment analysis tasks. They plotted unit values of negation, intensification, and concessive clauses which show interpretable differences in case of negation. They uniquely introduce methods for visualising a units saliency, how much a cell contributes to the final
meaning, which they determine through derivatives. Besides compositionality, visualisations can be used to make many more aspects of a model apparent.

Kádár et al. [20] used visualisation as an inspection method on an RNN trained for image captioning. Similar to Li et al they use a method that estimates the contribution of tokens to the final prediction and visualises it. Additionally they show methods that explore individual hidden layer units and show that there is specialisation towards the different tasks of image captioning. Notably they find hidden units that appear to specialise in memory for long term dependencies.

Besides tweaking their input, Linzen et al. [28] utilises visualisations to gain insights into the number-prediction task trained model. Activations are visualised per unit and interpreted superficially as having behaviours, such as: resetting on that but forgetting slowly, or remembering being in a relative clause indefinitely. In contrast to the previously mentioned studies they do not further interpret the results.

Attention

One mechanism that is often visualised for insights is the attention. As explained in the previous section, the attention mechanism allows for insights when inspecting trained models. These insights range from understanding what part of the input is important for the model, thus making it easier to spot mistakes in the model. In particular, visualising attention is useful for finding biases in the data, as well as general trends across examples or mistakes in the input.

In contrast to viewing attention on a few example sentences, the computation of co-occurrence matrices and plotting these can result in more insight into general tendencies of the model. The usefulness of visualisations decreases with the increasing dimensionality of networks and this makes it hard to draw quantitative conclusions.

Some statistics can be computed over attentions that do allow for more meaningful conclusions. One example of this is co-occurrence of high attention on certain words, this can be shown in a co-occurrence plot.

Visualising attention is an intuitive and effective tool for viewing the attention vectors. There are other vectors in neural networks that could potentially be visualised, such as activation in the hidden layer, however they cannot be directly related to input and output words making them a lot harder to understand. Other tools that can be used to analyse these deeper layers are explained below.

Diagnostic classifiers

Another method for going even deeper to gain insights which does give quantitative results are Diagnostic Classifiers [18]. Directly visualising values in attention, gates and other mechanisms can give insights, but it is hard for a human to check plots for thousands of examples and see what information is and isn’t there. Diagnostic classifiers extend the method of strategically choosing input and combines it with the values from these mechanisms. They are used to form and test hypotheses on the working of a model.

In practice, a model is “cut” at a certain layer or mechanism and a classifier is trained on the intermediate output that is the model’s state at that point. This could, but is not constrained to, cutting an encoder-decoder model in half, and attaching a classifier directly on the encoder. Effectively this classifier would classify over all the information available to the decoder and thus allows pinpointing of problems in the encoder. Mask’s or other types of labels can be generated that distinguish one aspect of input from the lack of it. If the Diagnostic Classifier can learn to predict the mask from the encoder, this means the encoder is saving that information in the hidden state. Intuitively one could say the model is sensitive to that aspect of the input. This is insightful for further development of the model: If the encoder never found the relevant information, tweaking of the decoder will never result in better performance.

Diagnostic Classifiers were originally introduced by Hupkes et al [18]. They argue that visualisation are often not enough for true insights into a models working. They introduce the approach on a compositional semantics tasks, the processing of nested arithmetic expressions, and show promising results in this domain. The process they used consists of first formulating multiple hypotheses on the information encoded and processed by the network. They derive prediction about features of the hidden state and train diagnostic classifiers to test the predictions.
Designing experiments with Diagnostic classifiers is time consuming, but very effective for hypothesis testing. By cutting straight into the deeper levels of your model they help with pinpointing weak points or otherwise interesting aspects of models.

Even deeper

To gain understanding of what role different units play in processing, Karpathy et al. [21] did an inspection of gate activations. They focus on the fraction of time that gates spend being left- or right-saturated (activations less than 0.1 or more than 0.9, respectively), which correlates to which aspects of the cell are sensitive to previous cells. Right-saturated update gate makes a model more sensitive to previous activation, whilst a model that is left-saturated across the board bases its output solely on current input.

Summary

First, we saw how dialogue systems went from more rule-based approaches to data-driven approaches and end-to-end systems. Our end-to-end dialogue systems were explained, starting with encoder decoder framework, LSTM’s, and attention mechanism. Additionally we discussed memory networks, an approach later used as a baseline. Lastly we discussed the characteristics dialogue data, such as how natural it is and how the data was obtained.

Second, we saw a general structure for disfluencies, as introduced by Levelt [26]. Additionally we described several subcategories of disfluencies, related to this structure, such as self-repair, hesitation, restart, and error-repair. Next we introduced several theories surrounding the parsing as disfluencies, such as the continuation problem. Noticeable is for example that the first word after the editing term position is crucial for determining how to parse a disfluency.

Last, we discussed several methods of analysis for neural models. Starting from traditional methods such as accuracy and error analysis, which are widely known and used. Continuing with visualisation and input altering methods which are less common yet still known in the machine learning field. Then we look at the relatively novel diagnostic classifiers. Classifiers are trained on parts of the model to determine sensitivities of these parts to specific phenomena.
CHAPTER 3

Experiments

In this chapter we dive into the main experiment of this thesis. We test a sequence to sequence model for its ability as a neural dialogue model in goal-oriented dialogue. We compare our models to memory networks, which are more often used as dialogue models and consequently have data to compare against. Besides availability of data points, memory networks are suited as a baseline since they are a neural model which is somewhat similar to our sequence to sequence approach with attention, as explained in the background section. In the comparison we have two objectives: Determine if the sequence to sequence model can do goal-oriented dialogue and determine if the sequence to sequence model generalises to data with unseen phenomena.

To achieve the first objective we focus on the bAbI tasks. The bAbI tasks are a small domain goal-oriented dialogue task which was designed specifically to test end-to-end dialogue systems. Memory networks are shown to perform these tasks well, testing whether our dialogue systems can perform on par with memory networks will benchmark our approach.

For the second objective we look into bAbI+ data. This is an extension of the first bAbI task with disfluencies. Memory networks are shown to fail at this task. Testing our model on this test will give insight into the similarity to memory networks and the nature of the difficulty with disfluencies.

Both tasks are set within a small domain and serve to fulfil a clear objective. This is called goal-oriented dialogue, which makes a more controlled and balanced environment to test the influence of disfluencies in. Additionally goal-oriented dialogue comes with a task which can be used in objective functions to train an end-to-end dialogue system with. The small size of the training set, 1000 dialogues, forces a dialogue systems trained on it to generalize relatively well, like humans do. In the first section we outline this data and the corresponding tasks performed with this data.

The sequence to sequence model we train is an encoder-decoder model using LSTM’s. Additionally the attention mechanism is used to overcome fading over long sequences. In the second section we discuss this model and its parameters. Additionally, we discuss the set-up of the experiments, the training procedure, and all aspects of the evaluation.

In the last section we present the results of both the parameter search, the first task, and the second task. We make some preliminary conclusions, which leads to the next chapter where we will discuss in-depth analyses done on the models obtained in the experiments described in this chapter.

1 Data

The goal of this experiment is to test a sequence to sequence dialogue model on its ability to do goal-oriented dialogue and on its ability to generalise to disfluencies. As a baseline we use memory networks which are more often used as dialogue systems. First we compare them to memory networks on a tasks which we know they can do well, after which we generalize to a feature memory networks were shown to perform bad at: disfluencies. In this section we explain
both these datasets and their baselines.

**Goal-oriented data**

End-to-end dialogue systems are not yet achieving desirable results on the unbounded domain of general dialogue. As was discussed in the background section, there are infinite topics one could discuss making dialogue hard to learn and train on. Complexity and unpredictability of this data is a problem for machine learning or data-driven approaches.

In contrast to unbounded domains are constrained or narrow domains. Here, the topic is specified beforehand and human participants are discouraged from deviating off-topic. Besides enforcing a narrow domain by restricting the topics, a narrow domain can be obtained more naturally. Goal-oriented dialogue naturally focuses on a smaller domain, namely the domain that the goal is about. Goal-oriented dialogue is dialogue where the dialogue serves an objective, such as making a reservation which stays in the domain that the reservation is to be made in. In this study we discuss goal-oriented dialogue and consequently the dialogues are set in a very narrow domain.

Another factor making unbounded dialogue hard is that it is very difficult to find an objective to train a dialogue system towards, multiple answers are possible and even if one could rank each output of a dialogue system by hand during training it would be hard to find non-subjective measures that could train a truly versatile chat bot. To omit these difficulties we focus on goal-oriented dialogue in this thesis.

A final advantage of focusing on goal-oriented dialogue is that some research has been done in this field, using different approaches but most notably similar to ours are memory networks. Notably are the bAbI tasks which are discussed below, which are specifically designed to test end-to-end dialogue systems on goal-oriented dialogue with a plethora of baselines.

**Unnatural data**

We are thus focusing on goal-oriented dialogue. The narrow domain this is set in is far less complex and involved as unbounded dialogue is. One factor in this is the source of the data. The dialogues in the data are human-machine dialogues, in contrast to human-human. Human-machine dialogue allow more control over certain phenomena such as the incremental phenomena we are testing our sequence to sequence model on. However, using human-machine data instead of human-human data greatly decreases the diversity and variability of the dataset, as stated by Williams and Young [48].

In contrast to this is the more involved and complex language used in human-human data, which comes at the expense of control over distributional and other properties. In this study we intend to compare a sequence to sequence model to memory networks on some toy tasks and then test their ability to generalise to disfluencies. To gain insight into this the small steps and deviations in the more artificial and controlled nature of human-machine dialogue allows for better understanding and quantification of our dialogue system’s performance.

A defining quality of dialogue data is the way a dialogue is generated or collected, where some methods results in more natural corpora then others. Often, the collection of data is done through the hiring of a human who is asked to perform a task by interacting with a dialogue system. One problem is that the hired humans are instructed in a certain way that may not actually coincide with the intentions and usage of the true user population. Another problem lies in the diversity of the population as it is hard to enforce diversity in the test population as you have diversity in a user population.

The bAbI tasks are generated through simulation which makes them even more controllable than human goal-oriented dialogue data. This direct tweaking of the phenomena helps with balancing datasets for training and additionally it makes inspection of the dialogue system a lot easier since it is easy to pinpoint the phenomena in the data.

**Disfluencies**

As stated above, beside testing a sequence to sequence model on goal-oriented dialogue abilities, we will be testing a sequence to sequence model on its ability to generalise to an unseen phe-
nomina in dialogue data. The phenomenon we focus on in this thesis is disfluencies. Disfluencies are literally disruption in fluency, in other words mistakes and other interruptions of the natural flow of dialogue. Disfluencies are very common in human dialogue, to facilitate the movement from bounded to unbounded dialogue and off-the-shelf use of dialogue systems in many different situations they should learn to handle disfluencies. On top of that, gaining insight in disfluencies is directly useful in troubleshooting generalisability abilities of dialogue systems.

Another reason for using disfluencies is the research done into disfluencies by Shalyminov et al. [38] which showed that neural dialogue systems cannot generalise to them. Their research provides us with a framework of expectations and a baseline to compare to in both their memory network approach as their grammar based approach.

In this section we discuss the bAbI tasks introduced by Bordes et al. [3]. Afterwards we go into the bAbI+ task which is introduced in a study by Shalyminov et al. [38] as discussed above. These two datasets together form the data that the dialogue systems introduced later in this chapter is trained and tested on, which is described in the rest of the chapter.

1.1 bAbI

The bAbI dialogue tasks were originally designed to test end-end goal-oriented dialogue systems [3]. The dialogues consist of human-machine interactions within the restaurant domain, where the end-to-end system is trained to utter all bot utterances correctly. The bot utterances include an API-call which summarizes the user’s information in a query language utterance, using slot’s for “cuisine”, “place”, “party size” and “price range”. Human utterances are generated through simulation, where goals are sampled from a distribution over all possible slot values. Natural language patterns are used to create utterances from these slot values. There are 43 patterns for humans and 20 different patterns for the machine. The machine will always reply the same way, since it reflects a narrow-domain rule-based dialogue model. Humans are more varied and can express each slot type in 4 different ways. Every dialogue consists of four to eight turns and combining the natural expressions with the samples user goal’s generates thousands of unique dialogues.

Since human utterances are generated through simulation the dialogues lack naturalness. The lack of naturalness allows for a more controlled testing environment which gives more precise indications of what aspects of dialogue dialogue systems can and can’t handle. Both for training and testing 1000 dialogues are provided which is relatively few, more is not desirable since dialogue systems should not require heaps of data for good results [3].

The bAbI dialogue tasks consist of 6 distinct tasks. Each task represents a different aspect of dialogue, the tasks are: Predicting an API-call; Updating an API-call; Displaying options; Providing extra information; and conducting full dialogue. Task 1-5 are displayed in Figure 3.1. The 6th bAbI task is adapted from dialogues used for the dialogue State Tracking Competition (DSTC2). States are used to generate API-calls and dialogues are shaped in the same format as other tasks. These dialogues were generated by actual humans talking to machines in contrast to the simulated conversations in the first tasks. This introduces problems such as mistakes due to speech recognition mishaps and other human errors. Examples dialogues of each task are included in Appendix B.

Williams and Young [48] argue against using rule-based systems as a gold standard for machine output: “While it would be possible to use a corpus collected from an existing spoken dialogue system, supervised learning would simply learn to approximate the policy used by that spoken dialogue system and an overall performance improvement would therefore be unlikely.” However the goal of the bAbI task is not improving upon the state of the art in a narrow domain such as restaurant reservations, but to test and benchmark the performance of end-to-end dialogue systems with no domain knowledge. Evaluating them on their ability to express themselves in both natural and query language. There is an example of bAbI+ dialogue included in Appendix B.

Baselines

To achieve this testing and benchmarking Bordes et al. present several baselines on the six tasks, each with different strengths and weaknesses. Comparing end-to-end dialogue systems to
Figure 3.1: Different bAbI tasks. Source: Bordes et al. [3]

Supervised embedding dialogue systems and a form of neural dialogue systems (memory networks with and without match type).

All baselines are evaluated through re-ranking, and consequently their output is a distribution over possible outputs. For the neural model this is achieved by training the model towards an output of a probability distribution over all possible outputs the model can generate based on the training data. The dialogue systems we train in this thesis do not output such a probability distribution but instead generate a sentence word by word from a vocabulary of words the model used. Even though this arguably makes the dialogue systems less comparable to baselines, generating is by definition harder than deciding.

1.2 bAbI+

Eshgi et al. [7] noticed that even though there is increase in goal-complexity across different tasks in bAbI, there is no increase in incremental lexical complexity which is integral to natural language. As discussed before naturalness is often low in human-machine dialogues. The jump from the unnatural dialogues obtained through simulation in task 1-5 to very natural dialogues that are directly transcribed from human utterances in task 6 is too big for end-to-end dialogue systems to bridge. This becomes evident through the drop in accuracy of baselines presented by Bordes et al. on this task [3]. To mitigate this gap in complexity and other aspects of natural data Eshgi et al. introduced bAbI+, an extension of the first task of bAbI with everyday incremental dialogue phenomena. An example dialogue of the bAbI+ dataset is shown in Figure 3.2

Three phenomena are introduced in the data: hesitations, restarts, and corrections. These
phenomena are added probabilistically through patterns, after which around 21%, 40% and 5% of the user’s turns contain corrections, hesitations and restarts, respectively. These percentages were chosen to reflect distribution of natural human dialogue. Modifications are in 11336 utterances in 3998 dialogue of six utterances on average, where each utterance can contain up to three modifications. The patterns used for the hesitations are designed such that lexical variation is constant, in other words no words are introduced that are not already in the original bAbI task.

The corrections introduced in bAbI+ are known as self-corrections, as explained in the disfluency section in the background chapter. The other two disfluencies, hesitations and restarts, are so-called forward-looking disfluencies. Below are examples of different disfluencies in the bAbI+ data where the underlined and red indicates the newly introduced sequence: a correction, a hesitation, and a restart respectively.

**Example 16.** I would like a french uhmn sorry a vietnamese restaurant

**Example 17.** we will be uhmn eight

**Example 18.** good morning uhmn yeah good morning

In respect to all of disfluency research, the disfluencies in bAbI+ are a small subset of possible subtypes. The dataset is still representative of the phenomena we are conducting research on, since it includes: Error-based self-corrections, hesitations, and repetitions. In bAbI+ repetitions are referred to as “restarts” and thus they referred to as restarts in the rest of this thesis. The bAbI+ dataset is purposefully designed to include fewer variation making the dataset more suitable for understanding the working of dialogue systems learning it.

**Baselines**

Eshghi et al. [7] use the bAbI+ dataset to test a grammar-based incremental dialogue phenomena in respect to a neural approach. They use memory networks -introduced by Bordes and Weston [3]- as a representative of neural dialogue systems in this experiment. They do additional research with memory networks on the bAbI+ task in another paper and the result presented in the paper by Shalyminov et al. [38] are considered baselines to this experiment. Shalyminov et al. find that the memory network cannot learn the task of bAbI+ achieving 28%, even though it can do first task of bAbI with 100% accuracy. By explicitly training on bAbI+ data, accuracy increases to 53% which is still far from perfect. This is surprising since memory networks can do other aspects of dialogue well, such as the original bAbI tasks. Shalyminov et al. then show that the grammar-based semantic parser they introduced achieves perfect scores and conclude that semantic knowledge is thus integral to solving the bAbI+ task. Important in this comparison is that rule-based dialogue models do not traditionally struggle with goal-oriented small domain dialogue and that the bAbI task do not challenge them as they challenged end-to-end dialogue.
systems. Shalyminov et al. do not test any additional dialogue systems or methods as baselines on the bAbI+ task.

2 Methods

2.1 Models

The dialogue systems implemented for this thesis are sequence to sequence models. Sequence to sequence models are known for their ability to generalize well even over long sentences. Since we use the entire dialogue as input, sequences tend to get quite long. Generalisation from bAbI to bAbI+ is a crucial part of the experiment, therefore sequence to sequence models are likely to perform well on the task at hand.

Two LSTM’s are used in an encoder-decoder framework. This theoretically separates the task of encoding the dialogue and thus parsing the disfluencies, from the decoding task were API-calls are generated. We do not train any distinct mechanism to help API-call prediction, the decoder is expected to learn both normal dialogue as API-calls, which have a distinct vocabulary.

Attention is used to help the model remember details that appear early in a long sequence. In other words, it helps remember things from the beginning of the dialogue. Additionally, attention will help with evaluation and inspection of the models.

Memory networks are also a sequence to sequence model and the hop mechanism can function as an attention mechanism. However, memory networks are more complex and specific, making generalisation harder. We use a more basic sequence to sequence approach which will be easier to analyse afterwards.

2.2 Training

For each task 1000 training dialogues, 1000 development dialogues and 1000 test dialogues are available. For task 1-5 an additional test set is available with out of vocabulary instances, also consisting of a 1000 dialogues. Whenever the bot speaks twice, this is denoted in the data as “<SILENCE>” as the human’s utterance.

The input sequence includes the memory of the rest of the dialogue. In practice this means that for in dialogue line $i$ the input is:

$$input_i = human_1 + bot_1 + human_2 + bot_2 + ... + human_i$$

$$output_i = bot_i$$

In figure 3.3 the input is thus the history, indicated in blue, and the $input_4$, indicated in green. The $output_i$ is indicated in red.

Models are trained to predict the next bot utterance given the entire history up to and including the previous human utterance. They are trained on all 1000 training dialogues of the bAbI or bAbI+ dataset. The models parameters are updated using the Adam optimizer.
Furthermore, minibatch is used with batches of size 32. The optimizer used is Adam [22] with a
learning rate of 0.001. The settings for hyper parameters are shown in table 3.1.

The dataset used is artificial and small which allows for over-fitting, the drop-out mechanism
[9] is known to help prevent this. This regularization method is implemented where input and
recurrent connections to LSTM units are skipped with a probability, in these models 0.2.

Table 3.1: Hyperparameters for all seq-to-seq models.

<table>
<thead>
<tr>
<th>Hyperparameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hidden layer size</td>
<td>500</td>
</tr>
<tr>
<td>Embedding size</td>
<td>128</td>
</tr>
<tr>
<td>Batch size</td>
<td>32</td>
</tr>
<tr>
<td>learning rate</td>
<td>0.001</td>
</tr>
<tr>
<td>trained for x epochs</td>
<td>20</td>
</tr>
</tbody>
</table>

We do a parameter search for hidden layer size to determine the best one to use for our
models.

2.3 Experimental Setup

We conduct two experiments, one to test the ability of the model to learn goal-oriented dialogue
and another to test the ability of the model to generalise to disfluencies in the data.

Experiment 1: The bAbI tasks

Before comparing sequence to sequence models to memory networks on their ability to generalize
from bAbI to bAbI+, we first need to establish whether or not they can solve the bAbI tasks.
To do this we evaluate them on the five tasks introduced by Bordes et al. [3] and the additional
sixth task consisting of DSTC2 dialogues.

For each of the six tasks both word accuracy and sequence accuracy are reported, as explained
above. We compare this to the memory network scores presented as a baseline by Bordes et al.
shown in Figure 3.4.

Experiment 2: bAbI+

In the second experiment we test how well the sequence to sequence model performs on task 1
with the bAbI+ data. Additionally this experiment assesses how well it can transfer from bAbI
to bAbI+ and vice versa.

Models were evaluated in a two-by-two set-up. This results in four conditions: bAbI/bAbI,
bAbI/bAbI+, bAbI+/bAbI+, and bAbI+/bAbI. Again both sequence and word accuracy is
reported. Five separate models are trained in all conditions and the average is reported.

Evaluation

For each experiment both word accuracy and sequence accuracy are reported. Word accuracy
is computed by counting the words that are correct in a sequence divided by the length of the
sequence, averaged over all sequences in a dataset. Sequence accuracy is computed by counting
all sequences that are correct in an entire set and dividing them by the number of sequences in
the entire set.

In the first experiment the different tasks give insight into the model’s ability to do different
aspects of goal-oriented dialogue. Consequently, sequence accuracy in these tasks represent
ability to do dialogue tasks.

In the second experiment every task is about API-call prediction. Here the sequence accuracies
represent API-call generation exclusively. The different conditions reflect how well the
model can learn and blindly generalise to disfluencies.
After reporting the scores several methods of introspection are used to gain understanding on how the models are solving the tasks.

First some example dialogues are looked at in detail. This can show patterns in errors and this can help in narrowing down which phenomena or which part of the task is performed on badly.

Second, attention is visualised. One of the mechanisms used by the models is attention, which allows for easy inspection. Visualising attentions can show strategies the model is using and this already allows some comparing to literature.

Third, analyses are performed to identify the role of the editing term, a construct from psycholinguistic theory.

Lastly, the model is analysed for incrementality in its processing and generating. The model is also analysed for sensitivity to other aspects of psycholinguistic theory on dialogue.

3 Results

3.1 Parameter Search: Hidden Layer Size

The optimal value for the hidden layer was determined empirically. First we ran different models trained on bAbI data where only the size of the hidden layer was varied. As can be seen in table 3.2 there is a big leap in accuracy at a hidden layer size of 500, which was consequently used in out further experiments.

In table 3.3 similar results are shown for models trained on bAbI+ data. A jump can be perceived at 500 hidden nodes again, 600 hidden nodes still slightly outperform 500 hidden nodes, however to keep complexity low and to make results more comparable 500 hidden nodes were used for bAbI+ data in further experiments.

One thing that seems curious is the sudden jump from 2.2 accuracy (out of 100) to 92.7 accuracy on sequence level from 400 to 500 hidden nodes. Word-level accuracy was already a lot higher, and sequences were likely only one word of, resulting in the big difference between word...
3.2 bAbI Tasks

First we test our model on all tasks in the bAbI dataset, results are shown in Table 3.4.

Table 3.4: Results of sequence to sequence model on different bAbI tasks. Sequence accuracy is reported with word accuracy in brackets

<table>
<thead>
<tr>
<th>Task</th>
<th>Sequence(Word)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1: Issuing API calls</td>
<td>100(100)</td>
</tr>
<tr>
<td>2: Updating API calls</td>
<td>100(100)</td>
</tr>
<tr>
<td>3: Displaying options</td>
<td>100(100)</td>
</tr>
<tr>
<td>4: Providing Info</td>
<td>57.2(93.6)</td>
</tr>
<tr>
<td>5: Full dialogues</td>
<td>100(100)</td>
</tr>
<tr>
<td>6: DSTC2</td>
<td>35.0(89.8)</td>
</tr>
</tbody>
</table>

Sequence to sequence models performed well on most tasks, showing a decline in Providing info and in task 6. The decline seen for task 4 may be explained by data sparsity, the model has to generate the information provided, such as phone numbers which are not seen that often during training. Task 6 is created using actual dialogues from the dstc2(dialog state tracking challenge 2). These dialogues feature more speech recognition errors and other mistakes which apparently throw the model off.

Accuracy on the fourth task ("Providing information") is lower than the other tasks, which may be due to sparsity of examples. The fourth task consists of giving extra information, such as phone numbers and street addresses based on the dialogue. These street addresses and phone numbers are given in a knowledge base which our model does not use during training or testing, explaining the low accuracy for the fourth task.

Comparing to memory networks

In the original paper introducing the bAbI tasks scores for memory networks were reported, see Figure 3.4.

As can be seen memory networks, the baseline for neural dialogue end-to-end dialogue models, have similar dips in performance. This makes sequence to sequence networks comparable to memory networks, which is not surprising since both have similarities in mechanisms used. Most importantly the models both perform perfectly on the first task of bAbI, which is the basis of the bAbI+ task and is thus crucial for comparing the networks results in the second experiment.
3.3 bAbI+ Task

We know sequence to sequence models can perform the first bAbI task well and on par with memory networks. Next we compare bAbI trained and bAbI+ trained model on bAbI test data and bAbI+ test data, similar to the set-up with memory networks in Shalyminov et al. [38]. In Table 3.5 the results achieved by Shalyminov et al. [38] with memory networks are shown are shown.

Table 3.5: Results for the bAbI+ experiment. Sequence accuracy is reported with word accuracy in brackets

<table>
<thead>
<tr>
<th>train / test</th>
<th>train accuracy</th>
<th>test accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>bAbI/bAbI</td>
<td>100</td>
<td>100(100)</td>
</tr>
<tr>
<td>bAbI+/bAbI+</td>
<td>100</td>
<td>99.9(99.99999)</td>
</tr>
<tr>
<td>bAbI/bAbI+</td>
<td>100</td>
<td>88.1(92.5)</td>
</tr>
<tr>
<td>bAbI+/bAbI</td>
<td>100</td>
<td>100(100)</td>
</tr>
</tbody>
</table>

Both the results for bAbI to bAbI and bAbI+ to bAbI+ are high, as expected. With regard to bAbI+ this shows that a sequence model can learn the task. The results on bAbI to bAbI+ show that indeed disfluencies pose a problem for the model when not trained on them. The results of bAbI+ to bAbI show that generalising is not a problem in itself, generalising to the simpler task happens without fail.

One question that we will try to answer in the next chapter is what the bAbI trained model can’t do, what capabilities is it missing to solve the bAbI+ task.

Comparing to memory networks

The results achieved by Shalyminov et al. [38] are shown in Table 3.6.

Table 3.6: Memory network API-call sequence accuracy from Shalyminov et al. [38] and API-call sequence accuracy for the seq-to-seq model.

<table>
<thead>
<tr>
<th>train / test</th>
<th>train accuracy</th>
<th>test accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>bAbI/bAbI</td>
<td>100</td>
<td>100.0</td>
</tr>
<tr>
<td>bAbI+/bAbI+</td>
<td>72</td>
<td>53</td>
</tr>
<tr>
<td>bAbI/bAbI+</td>
<td>100</td>
<td>50.4</td>
</tr>
<tr>
<td>bAbI+/bAbI</td>
<td>67</td>
<td>99</td>
</tr>
</tbody>
</table>

Similar to Shalyminov et al. we have 4 conditions. They reported scores for same-dataset accuracy and cross-dataset accuracy of Memory networks trained on bAbI and on bAbI+ data. Their results together with API-call accuracies for the seq-to-seq are shown in table 3.6. These surprising results indicate sequence to sequence models, with at least 500 hidden nodes, are complex enough for the bAbI+ task. Which models does the task better is indiscernible since both perform at ceiling level.

More surprising is that results indicate that sequence to sequence models perform better at learning the bAbI+ task, achieving a performance of 98.7% on bAbI+ to bAbI+ where Shalyminov et al. achieved 53% for their memory network set-up.

The second question we will try to answer in the next chapter is: How is the model that learned to do the bAbI+ task performing the task and is it consistent with the theory introduced in the background section of this paper? What mechanisms in the sequence to sequence models are crucial for performing this task and how are they used to solve the disfluencies?

Summary

In this chapter we described the main experiment of this paper. We discussed bAbI and bAbI+ data, the models used and the experimental set-up of the experiment.
First we saw that sequence to sequence model perform equally well as memory networks on the bAbI tasks, showing that indeed sequence to sequence models are similar to memory networks. In the second experiment we surprisingly saw that the sequence to sequence model can do the bAbI+ near perfect when trained on it, in contrast to memory networks. When generalising from bAbI data to bAbI+ data some error is introduced, but an accuracy of 81% is still achieved which is a lot higher than the 38% reported for memory networks. In the next chapter we will be diagnosing generalising from bAbI to bAbI+ and diagnosing the bAbI+ model.
In the experiment described in the previous chapter we trained a sequence to sequence model for the task of end-end goal-oriented dialogue in the restaurant domain. We tested a sequence to sequence model on its ability to do this task with disfluent data. Additionally, we tested the model trained on data without disfluencies (bAbI) on data with disfluencies (bAbI+) to see its generalising abilities. We start this chapter with an error analysis on this model trained on bAbI and tested on bAbI+. With this we attempt to highlight what aspects of disfluences are hard to capture, and what aspect of the task suffers from these disfluences.

When we trained on bAbI+ and tested on bAbI+, the model performed above expectations. In this chapter we analyse how it is performing this task so well. Visualising attentions is the first step in the process. This gives some insight into general patterns in the behaviour of the model, which results in hypotheses we test with additional analyses. We train a model without attention and examine the editing term in more detail by tweaking input. Lastly we train diagnostic classifiers to test incrementality of the model and to what extent it performs in line with disfluency theory. Many of the analyses discussed in this chapter were also reported in a joint paper with me, Hupkes, and Fernandez [17].

In the next chapter we discuss the results of both the experiments and the analyses laid out in this chapter.

1 Generalising to Disfluencies

Even though the model can learn to predict API-calls well from disfluent data, it is not performing as well without explicitly being trained on disfluent data. Accuracy drops when testing on bAbI+ data where disfluences are present, when the model is trained on bAbI data that does not contain disfluences. To identify which part of dialogue is affected most by the introduction of disfluences, we split the accuracy in Table 4.1. The accuracy is split into accuracy on API-calls and “other utterances” which denotes all utterances the bot makes that are not an API-call. This split clearly shows that the errors are mostly made when generating API-calls.

Table 4.1: Splitting the accuracy of bAbI/bAbI+ into API-calls and other utterances. Sequence accuracy is reported with word accuracy in brackets

<table>
<thead>
<tr>
<th></th>
<th>accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>full dialogue</td>
<td>88.1(92.5)</td>
</tr>
<tr>
<td>API-calls</td>
<td>50.8(90.2)</td>
</tr>
<tr>
<td>utterances</td>
<td>92.8(92.8)</td>
</tr>
</tbody>
</table>

API-calls are a summary of important content of the dialogue, errors in these reflect a lack of understanding of the meaning and content of the dialogue. In contrast, errors in other utterances which suggest a lack of dialogue abilities. For example wrongly generating “london” when “paris” is applicable is an error in understanding the dialogue contentually. In contrast to wrongly
generating "i’m on it" when a greeting was applicable, this indicates an lack of dialogue abilities. Since mistakes are mainly in API-calls, this suggests that disfluencies do not disrupt general dialogue ability but specifically hinder the model’s ability to understand the content of the entire dialogue and its ability to use it to predict an API-call. However, the bAbI tasks are synthetic, more complex dialogue actions may suffer from disfluencies.

Error Analysis

As concluded above most errors are made in API-calls, yet some errors are made in other utterances as well. To gain insights into what is causing there errors in the bAbI/bAbI+ set-up, a qualitative error analysis will be laid down below. By looking at cherry picked example dialogues we examine the workings of the model.

Mistakes in the dialogue are highlighted both where the source of the mistake is as where the occurrence of the mistake is, multiple examples are colour-matched (using red, green, and blue). This is not information provided to the dialogue systems at any time and only functions to help the reader with understanding of the examples.

Important to note is that mistakes in the dialogue cannot snowball further mistakes. The output of the model cannot impact the rest of the dialogue, since the human utterances are pregenerated together with the target bot utterances. For example when the model wrongly predicts something else than a greeting, the human will still reply as if he or she was greeted. Consequently dialogues shown below may look weird or unnatural.

```
good morning uhm yeah good morning

hello what can i help you with today

i’d sorry yeah i’d like to book a table in london

i’m on it

<SILENCE>

any preference on a type of cuisine

with uhm yeah with british cuisine

how many people would be in your party

for two no sorry four please

which price range are looking for

i am looking for a cheap restaurant sorry yeah i am looking for a cheap restaurant

ok let me look into some options for you

<SILENCE>

api_call british london two cheap

four

```

Figure 4.1: Example dialogue of bAbI model on bAbI+ data

The first dialogue, shown in Figure 4.1, starts off with a mistake made by the bot highlighted in red. Instead of greeting the customer, the bot immediately tells the user he is “on it”. In this examples lies a possible explanation for the missing 8% accuracy in the other utterances, utterances that are not API-calls. In this example one would expect the dialogue system to utter a greeting, instead it replies to the greeting with an utterance which is strictly used on turn 2: “I’m on it.” There are very few different greetings: hi, hello, and good morning. The disfluencies that are introduced, in this example restarts, significantly lengthen these greeting utterances, presumably causing the model to misinterpret the length of the input sequence as a second turn instead of a first turn. We ran into multiple of examples where the first bot utterance is disrupted and did not see examples were the model made mistakes in later turns. Additional experiments
into length shown later further confirm this intuition that the dialogue systems as a whole is not oversensitive to length, only in the first turn does length effect performance.

The second mistake in this example is made in the API-call, where the model wrongly predicts two. The blue marking indicates where the model made a mistake in interpretation, i.e. the source of the error. The mistake is a direct consequence of wrongly parsing a correction, showing that disfluencies do indeed disrupt the functioning of the model, which is an important assumption in our research. In previous experiments, both in the parameter search and in the final result, the difference between word and sequence accuracy was often really big. This API-call shows an explanation for this: only one word is off. Consequently the word accuracy is .8 while the sequence accuracy is 0, contributing to a big difference in the two types of accuracies.

In the second dialogue we examine, shown in Figure 4.2, a similar mistake is made. The model wrongly predicts cheap, which is indicated in red in dialogue. The utterance the model wrongly parsed is also indicated in red.

```
good morning

hello what can i help you with today

Can you book a table with french cuisine or no with british cuisine in london

in a uhm yeah can you book a table with british cuisine in london

in a cheap

price range no in a moderate price range for two

i'm on it

<SILENCE>

ok let me look into some options for you

<SILENCE>

api_call british london two cheap moderate
```

Figure 4.2: Example dialogue of bAbI model on bAbI+ data

Interestingly, no mistake is made in the generation of “british”. The utterance, very similar to the one discussed above, is marked in green. This suggests that the model does not make mistakes in all disfluencies. This could very well be due to the model randomly picking slot-words that fit the right category. As stated before, 21% of user turns contains a correction, on average each dialogue has 6 turns, meaning that on average there are 2 correction in every dialogue. If the model is randomly selecting slot-words we would thus expect one mistake per dialogue on average. The resulting word accuracy would be 75%, which in reality was 92%. It is difficult to predict the sequence accuracy since this greatly depends on the distributions of the mistakes over dialogues.

One factor that contributes to the higher success of the model is that if a restart and correction are both added in at the same place, the right slot word is mentioned twice while the wrong slot word is mentioned once. The example above with “french” and “british” shows this. This phenomena is not that common and thus this explanation is not sufficient. How the model is parsing these disfluencies and what is paying attention to when doing so is discussed in the oncoming section.

Another explanation for a higher than predicted word accuracy could be that some of the strategies useful for picking API-call slots for bAbI data are useful in bAbI+ data. For example the word “cuisine” appears only after words that could be in second slot, and in most correction, they will only appear in the end of the sentence.

Another example of a correct parse, without the repetition is present in the third and final example, shown in Figure 4.3. Here the model correctly picks “french” based on the utterance marked in green.

In this example the model picks wrongly twice as well, both “london” and “cheap” are predicted wrongly. These disfluencies and the corresponding API-call word are indicated in blue and red respectively. The first mistake (“london”) has a different structure than the second mistake, where “in” is repeated together with the price ranges. This suggests that the model is not doing one form of correction well while failing another. This partly contradicts the hypothesis
In conclusion, the model is mostly erring in API-calls. Only some disfluencies are predicted wrongly, most are guessed correctly. This is probably due to similarity in the data that is not disrupted by the disfluencies.

2 Attention

A core aspect of the encoder decoder framework, which makes them suited for predicting API-calls, is attention. In theory attention allows the model to pay attention to any input when producing specific words, which is particularly vital for the generation of API-call slots. In this section we first visualise attention to gain novel insights into the importance and use of attention. Next we further dive into the importance of attention by training models without attention and evaluating them on the same tasks.

2.1 Visualisation

As mentioned above attention is crucial to the working of the sequence to sequence models. A widely used method for inspecting such models is visualising this attention. Visualising attention effectively allows us to see which input the model attended for each output. Viewing visualisation of attentions will give insights into the inner workings of our sequence to sequence model.

In the visualisation in this chapter the output is on the y-axis and the input is on the x-axis. The colouring of the position in the graph shows the importance the model gives to each input for a given output. The darker the square the more important the input word is deemed for generating that output word. As discussed in the background attention is normalised to 1 hence each row in a plot sums to 1, columns are not subjected to this normalisation, since attention over different output words is independent.

Since attention is incorporating new words into the original attention vector at each input word, attention visualisations may show attention to a word by giving a high score to the word directly after the important word. In all written examples an underlined and red word indicates an error.

In this section we first further dive into the question of what about disfluencies makes handling of them hard without training on them. After that we examine attention visualisation on the same sentences of bAbI+ models, starting exploring how the model is doing this task. Last, we compare the attention visualisations of the different models on the same sentences to gain
some insights into what the disfluencies in data change about the attention usage. A handful of visualisations are discussed in this section, additional visualisation can be found in Appendix A.

bAbI model

First we examine the model discussed in the previous section: the model trained on bAbI data, data without disfluencies. The first visualisation is shown in Figure 4.4 and the corresponding dialogue is shown in Example 19.

Example 19. 

\texttt{H:good \ sorry \ yeah \ good \ morning}

\texttt{B:hello \ what \ can \ I \ help \ you \ with \ today}

\texttt{H:i’d \ uh\m \ yeah \ i’d \ like \ to \ book \ a \ table \ with \ \textit{japanese} \ oh \ no \ french \ \textit{cuisine} \ for \ \textit{four \ people} \ in \ a \ \textit{moderate \ price \ range}}

\texttt{B:i’m \ on \ it}

\texttt{H:<silence>}

\texttt{B:where \ should \ it \ be}

\texttt{H:in \ uh\m \ yeah \ in \ \textit{seoul} \ uh\m \ sorry \ madrid}

\texttt{B:ok \ let \ me \ look \ into \ some \ options \ for \ you}

\texttt{H:<silence>}

\texttt{B:<\texttt{API\_call british madrid six cheap}>}

Understandably, the model pays attention to the words that correspond to the words it is predicting. When it wrongly predicts “six” instead of “eight” it does this whilst paying attention to the word “six”. When correctly predicting “british”, it is paying attention to “food”, which, as explained in the introduction of this section, may be caused by the way sequence to sequence models integrate information into hidden state retaining most of the previous state’s information.

Figure 4.5: Visualisation of the bAbI decoder attention when generating the API call (vertical axis) for the disfluent dialogue in Example 20 (horizontal axis).
Example 20.  

H: good morning  
B: hello what can I help you with today  
H: can you make a reservation for six oh no eight in a uh yeah can you make a reservation for eight in a moderate price range no in a cheap price range in madrid  
B: i'm on it  
H: <silence>  
B: any preference on a type of cuisine  
H: i love british food  
B: ok let me look into some options for you  
H: <silence>  
B: <API_call> spanish london four cheap

Attention visualisation are more interesting when looking at the next example in Figure 4.5. The patterns in the attention vectors appear more chaotic and less interpretable. Even though the attention for “madrid” is straight-forward, the other attention patterns are dispersed. This suggests that even though attention is helping the process of API-call prediction, it is not the only mechanism that the model uses for this.

Most interesting is the attention in the final example we examine, shown in Figure 4.6.

Example 21.  

H: good morning  
B: hello what can I help you with today  
H: i’d like to book a table in paris uhm sorry london with sorry yeah i’d like to book a table in london with spanish food  
B: i’m on it  
H: <silence>  
B: how many people would be in your party  
H: for four people please  
B: which price range are you looking for  
H: in a cheap price range  
B: ok let me look into some options for you  
H: <silence>  
B: <API_call> spanish london four cheap

In this example the attention and prediction do not match. The model wrongly paid attention to “paris” even though it correctly predicted “london”. This confirms the hypothesis that other mechanisms are important in API-call prediction. We did not encounter this behaviour often, suggesting that attention is still important for generating API-call words.

bAbI+ model

Next we examined attention patterns of bAbI+ models. To increase comparability we look at patterns on the same sentences as were used before with bAbI attention visualisations. The visualisations are shown in Figure 4.7. Attention for bAbI+ models is more clean and clear, and very word is always correctly attended. In other words, when predicting “french”, the model is paying attention to “french”. As there are no mistakes this means the model is always attending the correct API-slot word.
Interesting to note is that the attention mechanism makes a distinction between corrections and restarts, which both mention the same slot-word twice. In a restart, shown in Figure 4.7c “four” is attended twice. In a correction, only the correct word is attended, even when the correct answer is mentioned again later. This can be seen in Figure 4.7a where eight is only attended once. This suggests that the model is differentiating between the different disfluencies, and possibly using principles from the disfluency theory described in the background to parse disfluencies. In the last section of this chapter we test this hypothesis with diagnostic classifiers.

A hypothesis that one could form is that the model somehow learns all possible slot-values that constitute a category; thus learning that “expensive”, “cheap”, and “moderate” are price ranges and thus attending the last word within that category when predicting a certain position. The attention visualisation in Figure 4.7a refutes this, as it is the “eight” within the correction structure that is attended, in contrast to the “eight” in the restart structure which is mentioned later. This hints at a more in-depth understanding of corrections and parsing of them learned by the bAbI+ model. Overall the bAbI+ attention patterns, seem more informed and correct than their bAbI counterparts, as if the introduction of disfluencies helped the model learn API-call prediction better.

Co-occurrence plots

The attention plots discussed in the previous section hint at a difference in how bAbI trained and bAbI+ trained models used attention. The attention of the bAbI model appears more random and less focussed. To see if we can quantify these qualitative results, co-occurrence plots were made.

Co-occurrence plots are created by plotting the average attention weight for each API-call word. For each API-call attentions are stored as a vector, after which averages are made per word, for example “spanish”. The resulting plot will have the word “spanish” after which it shows how much attention was paid to different words on average. This makes trends clear, however, a lot of details are lost. Whether two words are always both attended 50% or whether
one word is attended fully 50% of the time and the other word is attended fully the rest of times is undistinguishable. Words that contribute less than .2 to any target word are not shown, to keep the plot informative and understandable.

Figure 4.8: Attention for different API-slots for both models on bAbI+ data.

In line with the hypothesis formed by looking at cherry-picked attention plots, the bAbI model does not always attend to sensible words. The bAbI model is attending to the words “you” “with”, and “hello” with some frequency when predicting API-call words, which are not relevant to the API-call.

In Figure 4.6 we saw that the model wrongly attends the wrong form yet still predicts the right word. This behaviour does not show up in the co occurrence plot, suggesting that it is too infrequent to show up after averaging attentions.

The bAbI model frequently attends the word “<unk>”. This word is introduced by the way bAbI+ script solve the incorrect slot word in a pattern. Incorrect slot words are samples from a list of all possible slot values, this includes cuisine types and locations from the out of vocabulary (OOV) dataset. These words are not known to the bAbI model that did not see these during training. Attending <unk> suggests that the bAbI model has not learned the slot words by heart and is using surrounding words to predict the correct slot word. Additionally, this means bAbI is attending the wrong word, since <unk> is never the correct API-call as the bAbI+ procedure never alters the goal API-call.

Indeed bAbI+ attention is less diffuse and more in line with a clear method. The bAbI+ model never attends words such as “hello”, “with” or “you” and only attend to location words or words directly related to them. This suggests that the bAbI model has a better understanding of the meaning or use of the model. However, evidence is not conclusive for this hypothesis. It seems as if the introduction of disfluencies helped the model get a better understanding of the task and the words in the dialogues.

Interesting is that for some words the bAbI+ model attends the words directly after the target word. For example instead of learning to attend “french” and “italian” separately the word attended is food, which appears in either case. This in contrast with locations, where it attends each different city. This may be due to difference is variability of surrounding words in the data. However, this may be a side effect of how attention is computed as discussed previously. Attention is paid to a certain hidden state, which is in part build on previously occurring words. When paying attention to a certain state the model could also be responding to features of previous words still present in this state.

2.2 No Attention

We hypothesised that attention was crucial for performing the bAbI tasks. However, we could not find any evidence supporting that in our visualisations, in particular the co occurrence plots. This gave reason to suspect other mechanisms than attention are important for the bAbI trained model in predicting API-calls. To test this, we trained models for both tasks without the attention mechanism.
Methods

The set up is identical to the experiments described in Chapter 3. Models are trained and tested on bAbI and bAbI+ data in a two-by-two setup resulting in 4 different train/test combinations. We tested both trained models on three different versions of the test dataset: all sequences, API-calls, and other utterances. This results in twelve (3x4) evaluations. Reporting accuracy over all sequences allow results to be directly compared to our previous set up with attention. Reporting accuracy on only API-calls and all lines but API-calls allow us to more directly measure the influence on API-call generation which gives more direct insights into what aspect of the task the attention mechanism is crucial for.

Results

Results are shown in Table 4.2.

Table 4.2: Table for training without attention for both all sequences and only API-calls. Sequence accuracy is reported with word accuracy in brackets

<table>
<thead>
<tr>
<th>train/test</th>
<th>all sequences</th>
<th>API-calls</th>
<th>utterances</th>
</tr>
</thead>
<tbody>
<tr>
<td>bAbI/bAbI</td>
<td>83.5(95.3)</td>
<td>1.9(66.4)</td>
<td>100(100)</td>
</tr>
<tr>
<td>bAbI+/bAbI+</td>
<td>90.8(98.2)</td>
<td>18.4(80.6)</td>
<td>100(100)</td>
</tr>
<tr>
<td>bAbI/bAbI+</td>
<td>72.3(81.0)</td>
<td>0.00(58.7)</td>
<td>81.4(83.3)</td>
</tr>
<tr>
<td>bAbI+/bAbI</td>
<td>84.2(95.7)</td>
<td>20.425(81.404)</td>
<td>100(100)</td>
</tr>
</tbody>
</table>

The results clearly show that the attention mechanism is crucial to successful API-call prediction. Without attention accuracy becomes really low for API-calls. The high word accuracies show that it still recognises that it has to generate an API-call, but fails to fill in the slots correctly. The relatively high score on utterances show that the model’s ability to conduct dialogue is not impaired by the lack of attention.

When comparing the results to the model trained with attention, shown in Table 3.5, the model with attention is clearly better. Without attention the trends in the sequence and word accuracies stay the same, but the models completely fail all API-calls. Without the attention mechanism, results are still not comparable to memory networks, that do not struggle with API-call prediction without disfluencies. Interesting is that both the bAbI and the bAbI+ task falter when attention is not used. Instead of impairing mainly bAbI+ data where disfluencies are present. One explanation for this is that the dialogue history becomes quite long, making it hard for the sequence to sequence model to predict API-call slots mentioned in the beginning of the dialogue history. Since bAbI+ data is not particularly affected, this analysis does not teach us much on disfluency handling and tells us more about API-call prediction.

Overall attention seems crucial to API-call prediction. Surprisingly, the introduction of disfluencies appears to positively influence the directness of the attention mechanism.

3 Editing Term

In the previous section we saw that bAbI+ is not predicting the last occurrence of a certain category of words. Additionally we saw that the model’s attention appears to distinguish different types of disfluencies. In this section we utilise the analysis method introduced by Linzen on the bAbI+ trained model to inspect how it is performing its task and the implicit disfluency parsing. In the first section of this chapter we looked into the errors made by the system as a starting point for more in-depth analyses. A different but equally intuitive method for inspection is characterised by tweaking input with your phenomena to pinpoint how the phenomenon is handled by your model. Linzen himself tweaked sentences with a matching subject and a verb with variable words and distractors to test a models subject-verb agreement abilities.
3.1 No Editing Term

As described in the background section all disfluencies have an underlying structure. Three key components of a disfluency are the reparandum, the alteration, and the editing term. In this section we focus on the editing term.

In the bAbI task there is always an editing term present, even though in natural language this is not the case, and editing terms is only present in 18.520% of the disfluencies [16]. An example of a disfluency with editing term is shown in Example 22, where underlined and red denotes the editing term, and an example without an editing term is shown in Example 23.

Example 22. *I would like italian* no sorry *french food.*

Example 23. *for two ... for four people please.*

Because the editing term is not present in most disfluencies, this forces humans to learn more complex ways of detecting and differentiating disfluencies according to psycholinguistic theory [26] The first intuition we will test for is whether the model is basing its disfluency parsing too heavily on the editing term. We train and test the model on data where varying percentages of disfluencies have an editing term. If the model does indeed rely heavily on the editing term it will not generalise to data with less or no editing terms. If the model does generalise this indicates that other features of the data are used by the model to parse disfluencies.

Data generation

We created data with variable percentages of editing terms in disfluencies. The new datasets were created with code available in the original bAbI+ paper of Shalyminov et al. [38], which allows for tweaking of templates and thus editing terms used in all disfluencies.

In total we generate six datasets: train and test sets for three different percentages of editing terms. There is one train and test pair with no editing term in any of the correction or restarts, dubbed “noET”, there is one where there is an editing term in 20% of corrections and restarts dubbed “realET”, and there is the original dataset with an editing term in all corrections and restarts, dubbed “fullET”

Methods

The model is trained on each percentage of editing term and evaluated for each percentage of editing term. This results in a 3 by 3 set-up. Scores are reported for all combinations. The bAbI+/bAbI+ condition of the main experiment is the same as fullET/fullET in this experiment.

The training procedure and hyper parameters are the same as the experiment described in the previous chapter. In other words, the same objective function is optimised in the same way using the same parameters, such as hidden layer size or number of epochs. This makes results comparable to previous results.

Both sequence and word accuracy are reported. Again an average over five different executions of training are used for reliability of results.

Results

The results for this experiment are shown in Table 4.3. Results on test sets with less editing terms than the data set trained on are most informative towards our research question: Can the sequence to sequence model generalise to unseen or less seen phenomena.

Table 4.3: Results of all sequences with and without editing terms. Sequence accuracy is reported with word accuracy in brackets

<table>
<thead>
<tr>
<th>traindata</th>
<th>train acc.</th>
<th>noET acc.</th>
<th>realET</th>
<th>fullET</th>
</tr>
</thead>
<tbody>
<tr>
<td>noET</td>
<td>96.7 (99.5)</td>
<td>96.0 (99.3)</td>
<td>97.0 (99.4)</td>
<td>99.0 (99.8)</td>
</tr>
<tr>
<td>realET</td>
<td>96.9 (99.4)</td>
<td>95.4 (99.1)</td>
<td>96.5 (99.4)</td>
<td>99.5 (99.9)</td>
</tr>
<tr>
<td>fullET</td>
<td>100 (100)</td>
<td>97.9 (99.7)</td>
<td>98.1 (99.7)</td>
<td>99.9 (99.9)</td>
</tr>
</tbody>
</table>
The bAbI+ model performs surprisingly well, regardless of how many editing terms are in the data. Most surprising is the result for fullET/noET which indicates that even when training with all editing terms, it performs well when no editing terms are present. There is a performance drop of 6% for this condition. Additional experiments described below attempt to shed a light on the drop.

These results indicate that editing terms are not important for the functioning of the model. They do appear to help the model but the improvements are only minor.

### 3.2 Generalisation over Length of Editing Term

In the fullET/noET condition of the previous experiment with variable percentage of editing terms present we find a 6% drop in performance. To inspect what this can tell us about the models inner workings, we further analyse this by training on different lengths of editing terms. This experiment also tests the renowned generalisation abilities of seq2seq models.

Training on short editing terms and testing on long editing terms is a difficult environment for generalisation. The model has no indication during training that the lengths may vary, in a sense the model has to adapt without training. In theory this will limit the model in its adapting abilities since it cannot learn that lengths may vary. To counteract this, a testing scheme was designed, where the model is trained on for example short and long sentences and tested on a length that is not short, nor long, nor seen before. This allows the model to learn that length varies, without showing it specific examples of every length.

In the original experiment the editing terms shown in Example 24 are used in the patterns. The number in brackets indicates the length of the editing term. As you see only 1 or 2 words templates are used. These templates are denoted short in this experiment.

#### Example 24.

<table>
<thead>
<tr>
<th>Editing Term</th>
<th>Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>sorry</td>
<td>1</td>
</tr>
<tr>
<td>uhm sorry</td>
<td>2</td>
</tr>
<tr>
<td>oh no</td>
<td>2</td>
</tr>
<tr>
<td>no sorry</td>
<td>2</td>
</tr>
<tr>
<td>no</td>
<td>1</td>
</tr>
</tbody>
</table>

#### Data generation

To test how well the model generates to length two additional sets of interruptions were generated: medium and long. We created this data by systematically adapting bAbI data with different lengths of corrections. The scripts for bAbI+ data creation from Shalyminov et al. [38] were used.

The editing terms of medium(3-4 words) length used in the patterns are shown in Example 25. Patterns with editing terms of long length(5-6 words) are shown in Example 26.

#### Example 25.

<table>
<thead>
<tr>
<th>Editing Term</th>
<th>Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>uhm sorry uhm</td>
<td>3</td>
</tr>
<tr>
<td>no I really meant</td>
<td>4</td>
</tr>
<tr>
<td>oh I am sorry</td>
<td>4</td>
</tr>
<tr>
<td>no no no</td>
<td>3</td>
</tr>
<tr>
<td>sorry it should be</td>
<td>4</td>
</tr>
</tbody>
</table>

#### Example 26.

<table>
<thead>
<tr>
<th>Editing Term</th>
<th>Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>sorry uhm no sorry uhm</td>
<td>5</td>
</tr>
<tr>
<td>uhm no uhm I think uhm</td>
<td>6</td>
</tr>
<tr>
<td>ah let me rephrase to</td>
<td>5</td>
</tr>
<tr>
<td>no no no it should be</td>
<td>6</td>
</tr>
</tbody>
</table>

These templates and combinations of the sets are used to create six new datasets: short, mid, long, shortmid, shortlong, midlong. Note that the “short” dataset is the same as the original bAbI+ dataset. The dataset “shortmid” has an equal percentage of short and medium length editing terms.
Methods

We use the original condition of “FullET”. In other words, the datasets generated all have editing terms in every disfluency.

The six new datasets we generated as explained above, are used in 7 different evaluations, as illustrated in Table 4.4.

Table 4.4: Set up for editing term length-generalisation experiment

<table>
<thead>
<tr>
<th>train</th>
<th>test</th>
</tr>
</thead>
<tbody>
<tr>
<td>short + long</td>
<td>all</td>
</tr>
<tr>
<td>short + long</td>
<td>medium</td>
</tr>
<tr>
<td>short + medium</td>
<td>long</td>
</tr>
<tr>
<td>Medium + long</td>
<td>short</td>
</tr>
<tr>
<td>short</td>
<td>medium + long</td>
</tr>
<tr>
<td>Medium</td>
<td>short + long</td>
</tr>
<tr>
<td>long</td>
<td>short + medium</td>
</tr>
</tbody>
</table>

The models are trained similar to the original experiment and the same hyper parameters were used.

Again each evaluation is performed five times and the average sequence accuracy and word accuracy is reported. Accuracy is reported over all sequences, no distinction is made between API-calls and other utterances in this experiment.

Results

Results are shown in Table 4.5, where the scores displayed are sequence accuracies and the scores in brackets are word accuracies.

Table 4.5: Results of editing term length-generalisation experiment. Sequence accuracy is reported with word accuracy in brackets

<table>
<thead>
<tr>
<th>train/test</th>
<th>training accuracy</th>
<th>Testing accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>short/long/mid</td>
<td>100(100)</td>
<td>98.1 (99.7)</td>
</tr>
<tr>
<td>short/mid/long</td>
<td>100(100)</td>
<td>99.4 (99.8)</td>
</tr>
<tr>
<td>mid/long/short</td>
<td>100(100)</td>
<td>99.5 (99.9)</td>
</tr>
<tr>
<td>short/mid/long</td>
<td>100(100)</td>
<td>97.2 (98.9)</td>
</tr>
<tr>
<td>mid/short/long</td>
<td>100(100)</td>
<td>99.2 (99.7)</td>
</tr>
<tr>
<td>long/short/mid</td>
<td>100(100)</td>
<td>99.4 (99.8)</td>
</tr>
</tbody>
</table>

The first thing to notice is how close all scores are. The model shows great ability in generalising to unseen lengths. This indicates that it is not memorising editing terms and parsing disfluencies accordingly.

The results that are worst performing (short/midlong: 97.9) are nearly or fully equal to fullET/noET (97.9). Indicating that all knowledge gained from the editing term is related to the length of it, or memorising of the individual words.

Additional experiment reallylong

The model appears to do well on generalisation, to further test its abilities we also tested a shot in the dark approach as discouraged in the introduction of this experiment.

The set-up is very similar to the short/midlong condition in the previous experiment. Instead of generalising to slightly longer lengths, we now generalise to patterns where the editing term is 11-12 words long, which we call “reallylong”. The editing terms are shown in Example 27. These disfluencies surpass naturalness and mainly serve the purpose of testing the limits of our model.
Table 4.6: Results of bonus editing term length-generalisation experiment. Sequence accuracy is reported with word accuracy in brackets

<table>
<thead>
<tr>
<th>train/test</th>
<th>training accuracy</th>
<th>Testing accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>short/reallylong</td>
<td>100(100)</td>
<td>93.8(96.4)</td>
</tr>
</tbody>
</table>

The results of this experiment are shown in Table 4.6. Again sequence accuracy scores are depicted with word accuracy in brackets. Accuracy is still high but lower than the fullET/noET condition of the previous experiment. This is surprising since no editing term is intuitively less informative than a long editing term. This could be due to more utterances being disrupted due to length. Similar to the mistakes in the first utterance in bAbI/bAbI+ data where the bAbI model had only seen hello, good morning and other short utterances and was thus unable to recognise long greetings as greetings. Previous editing term lengths did not disrupt any other utterances, however the increased length may change this. Inserting an editing term of 12 words more than doubles the length of most sentences in the bAbI+ datasets.

One theory about bAbI+ models ability to do dialogue could be that the model answers based on the length of the sequence compared to lengths seen during training. This theory is supported by the failures in greetings seen in error analysis. The current results do not confirm this theory, since more failures would be expected when introducing 12 extra words in most of the sentences. This suggest the bAbI+ trained model has some other understanding of the utterances in the dialogue.

In conclusion, the bAbI+ model is resilient to tweaking of the editing term. Earlier results with attention suggested the model is recognising disfluencies, maybe even distinguishing between the different types. Editing terms are a distinct set of words, which makes them an easy way of detecting and distinguishing disfluencies. However, editing terms appear not crucial or necessary for the model’s working. This suggest that yet another heuristic of mechanism is doing the heavy lifting in the disfluency parsing department. In the next two sections we look into the encoder with more detail.

4 Incrementality

Incrementality is a well-established factor of human language abilities. Similar to the sequence to sequence model, humans perceive language incrementally, both word by word and sound by sound. Additionally, the build-up of semantic knowledge appears incremental. Neural models that exhibit incremental behaviour are interesting for cognitively plausible AI.

Dialogue is incremental in several ways: The turns are incremental, building up common ground and understanding one turn at a time. Utterances are also incremental within themselves, each word adds to the meaning of the previous incrementally. Disfluencies are not solely incremental, the reparandum is not recognisable until one sees an editing term or a conflicting word. Parsing a disfluency thus requires some “looking back”. To hypothesise on the disfluency parsing of the model, knowing whether it processes incrementally and how it processes is crucial to learn to what extend the system is similar to humans. To see if the model is processing incrementally we did three different experiments, described in this section.

4.1 Triggering API-calls

There is another way of tweaking input, in contrast to tweaking the input with the phenomenon in a way that would affect a human. This method consists of tweaking input in such a way that it utilises regularities in the data to trigger specific behaviour. In this last input-related experiment
we triggered an API-call at all utterances, to see to what extent the model is incrementally interpreting the dialogue.

Data is augmented such that every turn should induce an API-call. Since the model is evaluated sentence by sentence, this method cannot give insight into how the encoder processes words incrementally. Consequently this experiment focuses more on whether the entire model processes dialogue incrementally, turn by turn.

If processing is incremental one would expect similar accuracies to the accuracies achieved when predicting API-calls in the last turn, where they naturally occur, with some deviancy since there is repetition in the data. This repetition stems from the API-slot mentions in early turns, which are repeated in all following turns and are thus evaluated multiple times, thus counting heavier in the achieved accuracy.

Data generation

For this experiment, bAbI test data and bAbI+ test data is augmented with API-call triggering utterances. As explained before, in the main experiment data is augmented such that the input at turn n is the history up to that turn of both human and bot utterances, followed by the human utterance of turn n. The output at turn n is the bot utterance at turn n. In formulas this looks like:

\[
input_i = human_1 + bot_1 + human_2 + bot_2 + ... + human_i
\]

\[
output_i = bot_i
\]

At the end of every input the dialogue ending utterances are uttered: “ok let me look into some options <SILENCE>“. The output is generated based on the API-slots that have been mentioned until then. Any api-slot that is not yet mentioned is set to <unk>. In formulas this looks like:

\[
input_i = human_1 + bot_1 + human_2 + bot_2 + ... + human_i + “ok let me look into some options <SILENCE>”
\]

A more tangible example is given in Example 28, where you see the altered version of the first turn, the second turn and the final turn.

**Example 28.**

H: good morning ok let me look into some options <SILENCE>

B: api_call <unk><unk><unk><unk>

H: good morning

B: hello what can i help you with today

H: can uhm yeah can you book a table in a cheap price range in london ok let me look into some options <SILENCE>

B: api_call <unk>london <unk>cheap

... 

H: good morning

B: hello what can i help you with today

H: can uhm yeah can you book a table in a cheap price range in london

B: i’m on it

H: <SILENCE>

B: any preference on a type of cuisine

H: with french food

B: how many people would be in your party

H: for sorry yeah for six no for four please

B: ok let me look into some options

H: <SILENCE>

B: api_call french london four cheap

Methods

The bot is only evaluated for slots that are known, since it is unable to predict <unk> without training as it is not in the vocabulary. When evaluated for: “api_call french london <unk> cheap” , the model can score 0/3, 1/3, 2/3 or 3/3 When evaluated for: “api_call <unk> london
The model can score 0/1 or 1/1. Test sequences where all slots of the API-call or \(<\text{unk}>\) are skipped since the accuracy would be 0/0 and evaluating this gives no information at all. The previously trained models are used on the new test data. Again five different models were tested and averaged for both sequence and word accuracy.

**Results**

As can be seen in Table 4.7, the triggering of API-call prediction was not always successful, one in seven cases did not result in an API-call when feeding the augmented data to the bAbI+ model. Even though the model did not consequently predict API-calls, there are enough API-calls predicted to give an idea of the model’s inner working. The failed predictions may be caused by a different understanding of API-call prediction, causing the bAbI+ trained model to be less sensitive to the simple regularity of “ok let me look into some options \(<\text{SILENCE}>\)”.

<table>
<thead>
<tr>
<th></th>
<th>Accuracy</th>
<th>API accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>bAbI / bAbI</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>bAbI+ / bAbI</td>
<td>54.1</td>
<td>100</td>
</tr>
<tr>
<td>bAbI / bAbI+</td>
<td>66.6</td>
<td>66.6</td>
</tr>
<tr>
<td>bAbI+ / bAbI+</td>
<td>83.9</td>
<td>99.8</td>
</tr>
</tbody>
</table>

Besides the normal accuracies in the first column of the results table, another column displays scores for only utterances where an API-call was predicted. These accuracies are nearly identical to the accuracies in the original experiment. This suggests that the model is processing turns incrementally. If the model is not processing incrementally, one would expect accuracy to be very low when the model is forced to predict mid sentence, since it is only trained on understanding complete sentences. Crucial for this theory are the bAbI/bAbI+ accuracies which are similar at every utterance in comparison to at the final utterance. Showing that errors are made proportional at each utterance.

These results do not shed any light on whether within a turn the processing is incremental, this requires further inspection of the encoder LSTM’s of the models. An experiment involving diagnostic classifiers which focusses on this question is proposed later in this chapter.

### 4.2 Diagnostic Classifiers

Our previous analyses have resulted in valuable insights into the workings of a sequence to sequence LSTM dialogue system. Further understanding of the model’s performance requires going even deeper into the model’s inner working. Designing experiments for these deeper layers is hard since intuitive interpretation of inner representations of models is lacking. One method of working around this are diagnostic classifiers, as introduced in the background chapter. In this section we use diagnostic classifiers to dive into the encoder and first test it for incremental processing behaviour and lingering.

In the previous experiment we looked into whether or not the bAbI and bAbI+ models process incrementally. Both models appear to process incrementally. Earlier results showed that attention is crucial to the performance of this task. diagnostic classifiers are trained, whilst using the encoder LSTM, to predict the API-call incrementally. In contrast to the previous experiment which uses both the encoder and the decoder, this experiment focusses on the encoder LSTM. This allows us to locate where the “smart” behaviour is located.

**Methods**

We examine each slot separately. In other words, we first look at cuisines, then locations and so on. Each different slot value is represented by a number and the number -1 is used for padding. Padding is by default ignored for the evaluation, in other words -1 is used when no slot value is known or the evaluation of this slot value should not count towards accuracy that is currently computed. Input is viewed word by word, for each word the classifier has to predict the word
that is currently known for that slot: -1 if no word is mentioned and 0-9 respectively for the 10 different cuisines; 0-9 for the different locations; 0-2 for the three different price ranges; and 0-3 for the different party sizes. An example for price ranges is shown below in Figure 4.9. Any slot where no information is known is considered padding and the model is not scored or optimised based on what it predicts there.

```
with uhm yeah with british cuisine in a moderate no sorry a cheap price range
-1 -1 -1 -1 -1 -1 -1 -1 1 1 1 1 2 2 2
```

Figure 4.9: A labelled example sentence to evaluate whether models distinguish between mentions of different price ranges.

Models are evaluated in both sequence and word accuracy. Each condition is trained in five separate instances, which we take the average of.

**Results**

Results are shown in Table 4.8. The first thing to notice is the low accuracy. Clearly API-slots are not remembered from the first time they are mentioned to where they are to be reproduced often. This suggests the encoder is not actually building up an incrementally more complex representation.

Table 4.8: Accuracy incrementality diagnostic classifier for API-call generation.

<table>
<thead>
<tr>
<th>Slot</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cuisine</td>
<td>31.3</td>
</tr>
<tr>
<td>Location</td>
<td>25.9</td>
</tr>
<tr>
<td>Price range</td>
<td>57.3</td>
</tr>
<tr>
<td>Party size</td>
<td>43.0</td>
</tr>
</tbody>
</table>

There are multiple explanations for this. One explanation is that the model could be taking some time to parse the information, effectively shifting the understanding of the model forward in time. Another explanation is that the model recognises the API-call and remembers it for a short while, long enough to integrate it with a possible conflicting slot-word but not longer since the decoder will pick it up anyway with the attention mechanism. To distinguish between these additional experiments are required that shed a light on the incremental process of the model when encoding the utterance.

4.3 Progression Visualisation

In a second experiment diagnostic classifiers are used to look into the incrementality of the encoder in more detail. A diagnostic classifier is trained to predict the API-call slot for different location relative to the mention of the slot-word. This will show us how long the encoder keeps the API-call word in memory and thus shed some light on what the encoder is doing in the API-call prediction process.

**Methods**

Additional test and train sets are created focusing on specific relative positions. For each position the model is trained on data where everything is padding but the nth position from the API-slot. In bAbI+ data this may be multiple numbers per line. Examples of masks are shown in Figure 4.10.

**Results**

Results of this experiment are shown in Figure 4.11. At first glance, there is an overall downwards trend with all slot categories. Different API-slot words are remembered only temporarily, for 1-5 turns in 90% of the cases and in 3 and more turns in 50% of cases.
with uhm yeah with british cuisine in a moderate no sorry a cheap price range

<table>
<thead>
<tr>
<th>text</th>
<th>with uhm yeah with british cuisine in a moderate no sorry a cheap price range</th>
</tr>
</thead>
<tbody>
<tr>
<td>0th</td>
<td>-1 -1 -1 -1 -1 -1 -1 -1 -1 -1 1 -1 -1 -1 2 -1 -1</td>
</tr>
<tr>
<td>1st</td>
<td>-1 -1 -1 -1 -1 -1 -1 -1 -1 -1 1 -1 -1 -1 2 -1</td>
</tr>
<tr>
<td>2nd</td>
<td>-1 -1 -1 -1 -1 -1 -1 -1 -1 -1 1 -1 -1 -1 2</td>
</tr>
<tr>
<td>3rd</td>
<td>-1 -1 -1 -1 -1 -1 -1 -1 -1 -1 1 -1 -1 -1 -1</td>
</tr>
</tbody>
</table>

Figure 4.10: A labelled example sentence to evaluate how memory of API-slots progresses for price range and cuisine type.

Before, we hypothesised that the model could be taking some time to parse the information, effectively shifting the understanding of the model forward in time. These results dispute that hypothesis, as the prediction is near 100% on the first turns and progressively becomes worse over time. This suggests that the model recognises the API-call and remembers it for a short while, long enough to integrate it with a possible conflicting slot-word but not longer since the decoder will pick it up anyway with the attention mechanism. Evidence is not conclusive and further tests are done in the next section to determine whether the encoder is parsing the disfluency or the decoder is.

5 Disfluency Theory Diagnostic Classifiers

As seen in the previous chapter, the encoder model is not incrementally building up a more involved representation. The encoder is hypothesised to read the disfluency and remember it long enough to integrate it with a possible conflicting term and forget it afterwards since the decoder with attention will pick up the correct word and thus parses the disfluency. In this experiment we probe the encoder for knowledge of disfluency structure and sensitivity to disfluencies.
According to research, as discussed in the background chapter, disfluencies have a structure independent of meaning. In this set-up we probe the encoder’s representations for this structure of disfluencies. If the model is sensitive to the structure, we would expect it to be able to identify it. Since reparandum is not something one can solve incrementally and the classifiers are run incrementally, we do not expect the prediction of reparanda to have a high accuracy. In contrast we expect the accuracy on predicting alteration and editing terms to be high, if the model is sensitive to disfluencies and their structure.

Methods

The diagnostic classifier is trained with the encoder of the bAbI+ model. There are three distinct structure parts to be identified: alteration, editing term, and reparandum; which are indicated in Figure 4.12. A red one indicates a reparandum; a blue two indicates an editing term; and a green three indicates an alteration. Below, a hesitation and correction are shown, however the model is not taught to differentiate between the different types.

Figure 4.12: A labelled example sentence to evaluate whether models have distinct representations for reparanda, alterations, and editing terms.

The masks the diagnostic classifier is trained to predict are slightly different and do not contain all three structure items within one mask. During training on reparanda, all reparanda (both restarts and corrections) are marked as 1 and all other parts of the sentence are marked as zero. In the example above this means that all zeros, two’s and three’s are zeros. Two different masks are used for evaluations, one specific to corrections and one specific to restarts. For example in the correction test masks all reparanda of corrections are marked as 1, and correction of restarts are marked as padding. This allows us to differentiate accuracy for the two different disfluency types during testing. During training the models are trained on reparanda of all disfluency types, since the reparanda, editing terms and alterations are very similar to one another and the classifier can benefit in learning from the multitude of examples.

Similarly masks are generated for editing terms and restarts, more details are provided in the next paragraph. In total there are three training masks and six testing masks at the end of the mask generation procedure.

The resulting set-up is a 3 by 2 testing paradigm, with three different disfluency structure parts and two different disfluency types. For each evaluation both precision and recall are reported. As always each diagnostic classifier is trained five times and the average accuracies are reported.

Mask generation

The mask which marks the different disfluency categories mentioned above are created by automatically processing the previously generated bAbI+ data. In this procedure we exploited the regular nature of the synthetic dataset.

Each sentence is processed incrementally, and every editing term is identified, this includes restarts and corrections. The outer bounds of the reparandum are determined one one side by the beginning of the editing term and on the other side by the occurrence of the word or a similar word to the first word after the editing term. In other words, this process requires solving the identification of what the reparandum is.

To further illustrate the process we will re-examine an example from the background section, see Example 29. The editing term is underlined and indicated in red.

Example 29. I would like a french umm sorry a vietnamese restaurant

In this sentence the procedure first identifies the editing term. Based on the editing term we identify whether the disfluency is a hesitation which does not have any other identifiable parts.
Example 30. *I would like a french uhmm sorry a vietnamese restaurant*

For every editing term that is not part of a hesitation the surrounding reparandum and alteration are identified.

The next step is to identify the outer bounds of the reparandum. One bound of the reparandum is clearly the word prior to this, thus “french”. The other bound can be derived from the occurrence of “a” after the editing term. The user means to replace a previous mention of this word with the alteration hence the reparandum likely starts at the previous mention of this word. This last step is bold and marked blue in Example 31.

Example 31. *I would like a french uhmm sorry a vietnamese restaurant*

Simultaneously, the procedure finds the outer bounds of the alteration, the part that in some sense replaces the reparandum. One bound is obviously where the editing term stops, in this case “a”, bold and indicated in blue in Example 31.

The other bound has to be derived from “french” since french is a slot word, and repairs in the bAbI+ dataset always concern a slot-word, the procedure identifies the first occurrence of a similar slot word after the editing term. This last step is indicated in blue in Example 32.

Example 32. *I would like a french uhmm sorry a vietnamese restaurant*

The bAbI+ dataset features only three types of disfluencies and the structure of both restart and correction are very similar. This is exploited by the procedure when identifying reparanda and alterations. The difference between the two types of disfluencies when identifying its structure is that there are no changed slot-words involved. The bounds can be identified by exact copies of words as with the “a” example (shown above in Example 31).

Results

Results for this experiment are shown in Table 4.9.

<table>
<thead>
<tr>
<th></th>
<th>self-repairs</th>
<th>restarts</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>precision</td>
<td>recall</td>
</tr>
<tr>
<td>Reparandum</td>
<td>15.0</td>
<td>89.4</td>
</tr>
<tr>
<td>Editing term</td>
<td>37.3</td>
<td>99.4</td>
</tr>
<tr>
<td>Alteration</td>
<td>21.3</td>
<td>93.5</td>
</tr>
</tbody>
</table>

The general trend in the results is in line with expectations: Reparandum is predicted worst and editing term is predicted best. However, precision is very overall very low. This suggests that disfluency structure is not integral to the working of the encoder.

Interesting is that the editing term was not recognised successfully, despite it being a distinct class of words which are not used in the rest of the dialogue. Besides not needing the editing term for parsing disfluencies as we saw in the previous section, the model appears to barely notice editing terms.

Since the diagnostic classifier cannot identify the disfluency, this suggests that it is not the encoder making informed hidden states that have already parsed the disfluency. Instead it appears to be the decoder with its attention mechanism or another mechanism that is parsing the disfluencies and that is understanding the meaning behind the words.

Summary

When the model had to generalise to disfluencies without training, we saw that the bAbI model mostly struggled with API-call prediction. The attention mechanism proved to be integral to API-call prediction.

The bAbI model is not paying much attention to editing terms, however attention patterns suggest that it is paying attention to disfluency structure in general.
When looking at API-call prediction in more detail we get the impression that the encoder is not actually informed or “smart”. It appears to be used as a memory of sorts, were the decoder smartly picks what to attend.

Further experiments on disfluency structure confirm this hypothesis. Results show that the encoder is not very sensitive to disfluency structure, suggesting that the decoder is not only doing most of the API-call prediction and dialogue but also the disfluency parsing.
In this thesis we looked at analysing neural models. We inspected a sequence to sequence model on how it was dealing with a newly introduced linguistic phenomenon. We measured this influence on bAbI and bAbI+ data, both narrow domain goal-oriented synthetic dialogue data. The linguistic phenomena that we introduced are disfluencies: hesitations, corrections, and restarts. This phenomenon was previously tested by Shalyminov et al. [38] with memory networks and formal grammars, they found that memory networks could not do the tasks. We implemented a sequence to sequence model, a very similar neural architecture.

Sequence to sequence models are shown to perform well the first five bAbI tasks. Accuracy on the fourth task ("Providing information") is lower than the other tasks. The fourth task consists of predicting phone numbers and street addresses of restaurants, which are not seen that often. The sixth task, based on actual dialogues, is not performed well either, which is to be expected since the complexity is a lot higher. Besides complexity errors in speech recognition and other errors add noise to the already small dataset making it even harder to learn this task well.

Testing sequence to sequence models on dialogues with disfluencies when trained on dialogues with disfluencies shows that sequence to sequence models perfectly learn to parse disfluencies. This is expected since there are many structural commonalities in disfluencies, as explained in the background section. Sequence to sequence models are known to find patterns and similarities in data, explaining their success at parsing disfluencies.

Generalisation

We saw in the previous section that the model can process disfluencies well when trained on data containing them. Another interesting question answered is whether performance drops when introducing a phenomenon such as disfluencies without training with it. Our results show that disfluencies do not impair the ability of the model to conduct conversation much. There is a little decrease in accuracy on utterances that are not API-calls, but overall accuracy remains high. In contrast, serious losses are suffered in its ability to understand and summarise the content of the dialogue when self corrections are present, this is reflected in the badly predicted API-calls. API-calls do not suffer in form since mistakes made stay within domain, yet they do suffer in correct content. This suggests that sequence to sequence models do generalise to the new domain - with self-corrections- relatively well. They do not have any experience with choosing between multiple options and they do not, without training with examples, learn this well. However the model does learn disfluencies when trained on them, without explicit information on grammar or structure or even the existence of disfluencies.

For repetitions and restarts, no decrease in efficiency is perceived, and the bAbI trained model generalises to these forward-looking disfluencies without a problem. One explanation for this lies in the conflicting information introduced with corrections, that is not introduced by the other two disfluency types.
Memory networks

Surprisingly, both the model trained on bAbI data and the model trained on bAbI+ data greatly outperform the approach with memory networks introduced by Shalyminov et al. [38]. This despite the overwhelming similarities between memory networks and sequence to sequence models with the attention mechanism. Since we do not directly implement our own memory network it is difficult to determine what caused this difference in performance on the bAbI+ tasks.

Attention

Crucial in our model’s ability to do the bAbI task is attention. This becomes apparent when looking at the results of the sequence to sequence without attention, even though general dialogue abilities persist there is an immense decrease in accuracy on API-calls. Even though this is understandable this means that sequence to sequence modelling cannot parse the dialogue incrementally as humans appear to do. As explained below the model’s encoder acted more as a memory than an intelligent encoder that parses information from dialogue. This confirms the intuition that additional memory is required on top of linear processing, even though this memory need not be explicit as it is in memory networks.

Another interesting finding surrounding attention becomes apparent in the visualisation of the attention vectors. When comparing the attention visualisations of a bAbI trained model to the attention visualisations of a bAbI+ trained model, the bAbI+ attention appears less noisy and more clean. This suggests that instead of hindering understanding of dialogue, disfluencies in reality improve parsing and contribute to this understanding.

Incrementality

Results of triggering API-calls at every utterance indicate that disfluencies are processed incrementally by the decoder. Even though the metric failed at illiciting an API-call in some cases the accuracies still show the general trend of our main experiments. Further inspection of the encoder indicates that the encoder is not doing advanced processing of the information in the dialogue. Instead it appears to be simply processing the dialogue on a far more basic level. The decoder does more advanced processing with the attention mechanism and other functionalities.

Another side effect of incremental processing is lingering. Certain aspects of a sentence can linger beyond the mention, even when they are corrected. Even though the information does linger for a few words in the encoder, slot-words are not remembered until the end of a sentence which is sometimes required for information to be reused. This is not surprising since lingering is not promoted by the training procedure. The tasks in these experiments are narrow domain and have a single goal. This in contrast to human processing, where often multiple goals are achieved through dialogue at the same time. One experiment which would intuitively give the model motivation to learn to remember all slot-words is training a model towards two goals. During training the model is evaluated on both the original bAbI tasks as evaluated on predicting preferences of a customer based on the restaurant characteristics they mention or making the model suggest alternatives where all mentioned slot-words are considered correct answers. A more direct approach would be asking the model “how often was ‘italian’ mentioned” and doing multi-goal learning with that and the original bAbI objectives.

Human processing

The incrementality of the model contributes to the model’s cognitive plausibility, since human processing is known to be incremental. Proving some incrementality in the models trained in this thesis, indicates similarity to human processing.

To further extend the similarity to humans, diagnostic classifiers could be combined with neural imaging data. Measuring human activation when solving disfluencies and testing a model on disfluency data. A diagnostic classifier can then be trained to predict human brain activity based on the hidden activation of the model. If there is a correlation for the classifier to pick up on, that indicates similarity in approach or in processing of the data.
Inspecting models for hypotheses formulated based on research into human processing of the phenomena is important in this research. This gives novel insights in this thesis. Even though a full understanding is still incomplete, the black box is opened at least partially.

**Disfluency structure**

The diagnostic classifier experiment for API-call generation shows that the encoder is not doing complex parsing. Instead it is functioning as a memory. It is unclear from this experiment whether the encoder is incorporating disfluency parsing or whether the decoder is parsing disfluencies when paying attention to the encoder states. The final diagnostic classifier experiment sheds some light on this, as its results indicate that it is the latter. The encoder is not sensitive to disfluency structure, which it would need to integrate the disfluency into it’s hidden state.

Attention visualisations indicate that the model (encoder and decoder combined) is distinguishing between different types of disfluencies. However it is not doing this by learning the editing terms, as it still has high accuracy on the task when they are not present. This suggest that the decoder is using disfluency structure to parse the disfluencies. Additional experiments are required to identify how the decoder is performing this parsing, and whether the decoder is sensitive to disfluency structure or not.

Overall, sequence to sequence models appear very promising for generalisation problems. Inspecting the models proved easier when using human processing as a starting point. This gave insights into the model’s workings and this is also a good starting point for cognitively plausible NLU models.
List of Figures

2.1 LSTM layout, source: stratio .............................................. 9
3.1 Different bAbI tasks. Source: Bordes et el.[3] ............................ 21
3.2 Example of bAbI+ dialogue. Green: restart, blue:hesitation, red:correction. .. 22
3.3 Example dialogue of Task 1, colours indicate inputs for turn 4 ............... 23
3.4 Baseline results on bAbI tasks. Source: Bordes et al. [3] ..................... 26
4.1 Example dialogue of bAbI model on bAbI+ data ............................. 30
4.2 Example dialogue of bAbI model on bAbI+ data ............................. 31
4.3 Example dialogue of bAbI model on bAbI+ data ............................. 32
4.4 Visualisation of the bAbI decoder attention when generating the API call (vertical axis) for the disfluent dialogue in Example 19 (horizontal axis). ........ 33
4.5 Visualisation of the bAbI decoder attention when generating the API call (vertical axis) for the disfluent dialogue in Example 20 (horizontal axis). ........ 33
4.6 Visualisation of the decoder attention when generating the API call (vertical axis) for the disfluent dialogue in Example 21 (horizontal axis). ............ 34
4.7 Visualisation of the bAbI+ decoder attention when generating the API call (vertical axis) and a dialogue (horizontal axis). .......................... 35
4.8 Attention for different API-slots for both models on bAbI+ data. ........... 36
4.9 A labelled example sentence to evaluate whether models distinguish between mentions of different price ranges. ............................. 44
4.10 A labelled example sentence to evaluate how memory of API-slots progresses for price range and cuisine type. ............................. 45
4.11 Accuracy at position relative to mention of each type of slot ............... 45
4.12 A labelled example sentence to evaluate whether models have distinct representations for reparanda, alterations, and editing terms. ...................... 46
## List of Tables

<table>
<thead>
<tr>
<th>Table</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>General form of a disfluency [26]</td>
<td>11</td>
</tr>
<tr>
<td>3.1</td>
<td>Hyperparameters for all seq-to-seq models</td>
<td>24</td>
</tr>
<tr>
<td>3.2</td>
<td>Hyper-parameter search for hidden layer size for seq-to-seq model trained on bAbI. Sequence accuracy is reported with word accuracy in brackets.</td>
<td>25</td>
</tr>
<tr>
<td>3.3</td>
<td>Hyper-parameter search for hidden layer size for seq-to-seq model trained on bAbI+. Sequence accuracy is reported with word accuracy in brackets.</td>
<td>25</td>
</tr>
<tr>
<td>3.4</td>
<td>Results of sequence to sequence model on different bAbI tasks. Sequence accuracy is reported with word accuracy in brackets.</td>
<td>26</td>
</tr>
<tr>
<td>3.5</td>
<td>Results for the bAbI+ experiment. Sequence accuracy is reported with word accuracy in brackets.</td>
<td>27</td>
</tr>
<tr>
<td>3.6</td>
<td>Memory network API-call sequence accuracy from Shalyminov et al. [38] and API-call sequence accuracy for the seq-to-seq model.</td>
<td>27</td>
</tr>
<tr>
<td>4.1</td>
<td>Splitting the accuracy of bAbI/bAbI+ into API-calls and other utterances. Sequence accuracy is reported with word accuracy in brackets</td>
<td>29</td>
</tr>
<tr>
<td>4.2</td>
<td>Table for training without attention for both all sequences and only API-calls. Sequence accuracy is reported with word accuracy in brackets</td>
<td>37</td>
</tr>
<tr>
<td>4.3</td>
<td>Results of all sequences with and without editing terms. Sequence accuracy is reported with word accuracy in brackets</td>
<td>38</td>
</tr>
<tr>
<td>4.4</td>
<td>Set up for editing term length-generalisation experiment</td>
<td>40</td>
</tr>
<tr>
<td>4.5</td>
<td>Results of editing term length-generalisation experiment. Sequence accuracy is reported with word accuracy in brackets</td>
<td>40</td>
</tr>
<tr>
<td>4.6</td>
<td>Results of bonus editing term length-generalisation experiment. Sequence accuracy is reported with word accuracy in brackets</td>
<td>41</td>
</tr>
<tr>
<td>4.7</td>
<td>Results of incremental API-call generation</td>
<td>43</td>
</tr>
<tr>
<td>4.8</td>
<td>Accuracy incrementality diagnostic classifier for API-call generation.</td>
<td>44</td>
</tr>
<tr>
<td>4.9</td>
<td>Results of the disfluency structure diagnostic classifiers experiment</td>
<td>47</td>
</tr>
</tbody>
</table>
Bibliography


Appendix A: Attention Visualisations

In this appendix are additional attention visualisation from both models (as indicated in the caption below the figure) when evaluated on bAbI+ data.

Figure 5.1: Attention visualisation for bAbI trained model.
Figure 5.2: Attention visualisation for bAbI trained model.
Figure 5.3: Attention visualisation for bAbI+ trained model.
Figure 5.4: Attention visualisation for bAbI+ trained model.
Appendix B: Example Dialogues

Below are example dialogues of all bAbI+ (1) and bAbI (6) tasks. Each line begins with a number, other new lines are not in the data but only function to keep content on the page.

bAbI+

1 hi uhm yeah hi hello what can i help you with today
2 i'd like to book a table with japanese no british cuisine i'm on it
3 <SILENCE>where should it be
4 in sorry yeah in uhm bombay how many people would be in your party
5 for four oh no for two please which price range are looking for
6 in sorry yeah in a expensive price range please ok let me look into some options for you
7 <SILENCE>api_call british bombay two expensive

bAbI task1

1 hi hello what can i help you with today
2 can you book a table i'm on it
3 <SILENCE>any preference on a type of cuisine
4 i love italian food where should it be
5 in paris how many people would be in your party
6 we will be two which price range are looking for
7 in a cheap price range please ok let me look into some options for you
8 <SILENCE>api_call italian paris two cheap
9 you rock you're welcome

bAbI task2

1 hi hello what can i help you with today
2 may i have a table with italian food for four in a cheap price range in rome i'm on it
3 <SILENCE>ok let me look into some options for you
4 <SILENCE>api_call italian rome four cheap
5 actually i would prefer for eight people sure is there anything else to update
6 actually i would prefer in a expensive price range sure is there anything else to update
7 no ok let me look into some options for you
8 <SILENCE>api_call italian rome eight expensive
9 you rock you're welcome
5 actually i would prefer for eight people sure is there anything else to update
6 actually i would prefer in a expensive price range sure is there anything else to update
7 no ok let me look into some options for you
8 <SILENCE>api_call italian rome eight expensive
9 you rock you’re welcome

bAbI task3

1 resto_rome_cheap_indian_6stars R_phone resto_rome_cheap_indian_6stars_phone
2 resto_rome_cheap_indian_6stars R_cuisine indian
3 resto_rome_cheap_indian_6stars R_address resto_rome_cheap_indian_6stars_address
4 resto_rome_cheap_indian_6stars R_location rome
5 resto_rome_cheap_indian_6stars R_number eight
6 resto_rome_cheap_indian_6stars R_price cheap
7 resto_rome_cheap_indian_6stars R_rating 6

bAbI task4

1 resto_rome_moderate_spanish_1stars R_phone resto_rome_moderate_spanish_1stars_phone
2 resto_rome_moderate_spanish_1stars R_cuisine spanish
3 resto_rome_moderate_spanish_1stars R_address resto_rome_moderate_spanish_1stars_address
4 resto_rome_moderate_spanish_1stars R_location rome
5 resto_rome_moderate_spanish_1stars R_number six
6 resto_rome_moderate_spanish_1stars R_price moderate
7 resto_rome_moderate_spanish_1stars R_rating 1
bAbI task5

1 good morning hello what can i help you with today
2 i’d like to book a table for two in london in a moderate price range i’m on it
3 <SILENCE>any preference on a type of cuisine
4 with spanish cuisine ok let me look into some options for you
5 <SILENCE>api_call spanish london two moderate
6 instead could it be with british food sure is there anything else to update
7 no ok let me look into some options for you
8 <SILENCE>api_call british london two moderate
9 resto_london_moderate_british_5stars R_phone
10 resto_london_moderate_british_5stars R_cuisine british
11 resto_london_moderate_british_5stars R_address
12 resto_london_moderate_british_5stars R_location london
13 resto_london_moderate_british_5stars R_number two
14 resto_london_moderate_british_5stars R_price moderate
15 resto_london_moderate_british_5stars R_rating 5
16 resto_london_moderate_british_3stars R_phone
17 resto_london_moderate_british_3stars R_cuisine british
18 resto_london_moderate_british_3stars R_address
19 resto_london_moderate_british_3stars R_location london
20 resto_london_moderate_british_3stars R_number two
21 resto_london_moderate_british_3stars R_price moderate
22 resto_london_moderate_british_3stars R_rating 3
23 resto_london_moderate_british_2stars R_phone
24 resto_london_moderate_british_2stars R_cuisine british
25 resto_london_moderate_british_2stars R_address
26 resto_london_moderate_british_2stars R_location london
27 resto_london_moderate_british_2stars R_number two
28 resto_london_moderate_british_2stars R_price moderate
29 resto_london_moderate_british_2stars R_rating 2
30 <SILENCE>what do you think of this option: resto_london_moderate_british_5stars
31 i love that great let me do the reservation
32 what is the phone number of the restaurant here it is
33 thank you is there anything i can help you with
34 no thank you you’re welcome

task 6

1 <SILENCE>Hello , welcome to the Cambridge restaurant system . You can ask for restaurants by area , price range or food type . How may I help you ?
2 <SILENCE>What kind of food would you like?
3 persian restaurant in the north part of town I’m sorry but there is no restaurant serving persian food
4 chinese food Could you please repeat that?
5 chinese food Did you say you are looking for a restaurant in the north of town?
6 yes There are restaurants serving chinese in the north of town . What price range would you like?
7 any api_call chinese north R_price
8 the_hotpot R_post_code the_hotpot_post_code
the_hotpot R_cuisine chinese
the_hotpot R_location north
the_hotpot R_phone the_hotpot_phone
the_hotpot R_address the_hotpot_address
the_hotpot R_price expensive
the_hotpot R_rating 9
hakka R_post_code hakka_post_code
hakka R_cuisine chinese
hakka R_location north
hakka R_phone hakka_phone
hakka R_address hakka_address
hakka R_price expensive
hakka R_rating 1
golden_wok R_post_code golden_wok_post_code
golden_wok R_cuisine chinese
golden_wok R_location north
golden_wok R_phone golden_wok_phone
golden_wok R_address golden_wok_address
golden_wok R_price moderate
golden_wok R_rating 7
<SILENCE>the_hotpot is a nice place in the north of town serving tasty chinese food
<SILENCE>golden_wok is a nice place in the north of town serving tasty chinese food
phone number golden_wok is a great restaurant
phone number The phone number of golden_wok is golden_wok_phone
thank you good bye you are welcome