MSc Artificial Intelligence  
Track: Natural Language Processing

Master Thesis

External Memory Enhanced Sequence-to-Sequence Dialogue Systems

by

JACOB VERDEGAAL

5677688

June 21, 2018

30 ECTS  
February 2017 – June 2018

Supervisor:  
Dr R FERNÁNDEZ ROVIRA
Dr E BRUNI

Assessor:  
Dr J ZUIDEMA

ILLC
In Natural Language Processing the sequence-to-sequence, encoder-decoder model is very successful in generating sentences, as are the tasks of dialogue, translation and question answering. On top of this model an attention mechanism is often used. The attention mechanism has the ability to look back at all encoder outputs for every decoding step. The performance increase of attention shows that the final encoded state of an input sequence alone is too poor to successfully generate a target. In this paper more elaborate forms of attention, namely external memory, are investigated on varying properties within the range of dialogue.

In dialogue, the target sequence is much more complex to predict than in other tasks, since the sequence can be of arbitrary length and can contain any information related to any of the previous utterances. External memory is hypothesized to improve performance exactly because of these properties of dialogue. Varying memory models are tested on a range of context sizes. Some memory modules show more stable results with an increasing context size.

I thank my supervisors for their ideas, time and guidance by great feedback, without which I would not have been able to complete this report. Furthermore, I thank my family for support and encouragement to continue, even in hopeless times.
1 Introduction

One of the unique human traits is the ability to converse with each other, which is an artifact of our intelligence. In the field of Artificial Intelligence (AI) much research has been done to model this unique behavior. A natural conversing digital system has long been the holy grail in AI, encouraged by the Turing Test and the Loebner prize. Although the test was initially proposed as an indication of intelligence, the superficial challenge of creating a naturally conversing system has proven to be hard by itself.

In the past decade some systems are claimed to have passed the test, but not all AI researchers agree, due to various reasons. For example, the computer program ‘Eugene Goostman’ has been claimed to pass, but this is debated based on the setting of the test with the following argumentation. The program is presented as a 13 years old boy, and thus naturally viewed upon as a bit clumsy. Furthermore, it passed with just enough successes. Another example is cleverbot, this system has an online text-chat interface which people can use to converse with it. The system saves all human contributions, and draws from these to generate replies, making sure cleverbot’s contributions resemble natural utterances. But assuming human conversational contributions carry information which is crafted by the speaker for that particular contribution, cleverbot’s contributions are only superficial natural, since it can only draw on utterances it has already seen and can thus only convey content from this limited set, and is therefore not natural in respect to content.

Furthermore, the Turing test itself is debated as being a test for system intelligence and previous observation brings forth the following proposition. The ability of a system to chat like a human, but doing so without reasoning on content, is what I would like to call the superficial task of the Turing test or dialogue in general. The more sophisticated task of creating content is much harder and therefore under investigation in AI and this thesis in particular.

The goal of this work is not to develop a dialogue system to compete in the Turing-test, but to investigate techniques which can be added to state-of-the-art end-to-end models in a straightforward manner to improve content of replies of a dialogue agent. Furthermore, since relevant information is often found in utterances further back in the dialogue history than the three preceding ones, which are commonly used in research, emphasis in this work lies on the number of utterances to be used as input.

Although, the state-of-the-art models in closed domain tasks, such as telephonic reservation systems and web customer assistants, helping users navigate FAQ’s are successful on the sophisticated task. But exactly because the domain is restricted and the systems are specifically adapted to these domains, the systems perform well by exploiting the possibility of making use of hard-coded rules to navigate the domain knowledge. When trying to converse on topics outside the domain, the systems fail
because hard-coded rules are task specific and thus these systems are not eligible to pass the superficial part of the Turing test.

In the past few years there have been great advancements in the field of Natural Language Processing (NLP), which are all based on the sequence-to-sequence (seq2seq) \textsuperscript{33} or encoder-decoder \textsuperscript{6} Neural Network (NN) models. These networks are able to model language. A language model defines a probability distribution over a set of words given \( n \) previous words. A good model only generates grammatical and coherent sentences. Since language is technically a set of sequences (sentences made up of words) of varying length, the same network is used for every word. Before explaining the specifics of this approach, a brief description of the basics of NNs follows.

Artificial NNs are inspired on the neural networks that form our brain. The base component is an artificial neuron, first introduced by McCullogh and Pitts in 1943.\textsuperscript{7} It receives multiple numerical inputs which are summed and mapped to a single output value by an activation function. A NN is a set of artificial neurons connected in any way, capable of receiving input and generating output. The amount of possible architectures of NNs is infinite, but some have proven to be useful for particular tasks.

The aforementioned NNs used in NLP are able to process, or encode, sequences of words (represented as a row or vector of numbers) and yield a single vector representing the entire (encoded) sequence. Input sentences vary in length and are processed by the same layer, which is composed of a set of identical artificial neurons (or nodes) with only in- and output edges. The output of this layer for word \( i \) is used as additional input in processing word \( i + 1 \), thereby modeling the dependencies between words. Long term dependencies, spanning multiple (unrelated) words can be kept in internal memories the nodes have. Models using this setup are generally referred to as Recurrent NNs (RNNs), and are extensively described in section \textsuperscript{2.3.2}. The last vector is used to predict an answer or as start-state to generate a sequence of words. This process is often referred to as decoding \textsuperscript{20}. Since the dependencies between words can span a range (much) larger than one, the memory can be queried when needed. This memory is in this work assumed to be responsible for language modeling.

Another milestone in these advancements was the introduction of the attention mechanism \textsuperscript{4} which checks intermediate states of the encoder at every step in decoding. It has proven to perform well in many NLP tasks such as reasoning \textsuperscript{26}, question answering \textsuperscript{39} and translation \textsuperscript{18} and has become a standard addition on recurrent encoder-decoder models. Recently this approach has proven itself in dialogue as well \textsuperscript{37}. This conversational model from 2016 can learn to be a help-desk employee and reset a password, or mimic an educated psychiatrist as exemplified in the paper, and is claimed not to be restricted to a particular domain, but by content in the training data only.

This system uses a standard seq2seq setup, thus showing content can be learned by the internal memory. Past years alternative forms of attention are developed, often referred to as ‘external’ memory. Essentially, external memory is a modification of the basic attention mechanism, since it stores intermediate outputs while encoding and in-

\textsuperscript{7}See Wikipedia
cludes them in the computational path (while basic attention does not). In this work I will test two distinct variants. One, the Neural Semantic Encoder \cite{22, 23}, uses the attention mechanism in encoding stage as well. The attention mechanism uses semantic similarity and hence the name. This memory type initializes the memory with the to be encoded sequence. The other, the Differentiable Neural Computer \cite{10}, starts with an empty memory and applies additional data structures to encode dependencies between parts of the memory. The technical specifications are given in section 3.

Above all, when viewing language in the frame of dialogue, sequence processing is a process of information accumulation. All data up to that point in the sequence of sentences (the context) is relevant to generate the next. Recent work has shown that using encoded context can benefit reply generation by modeling the contributions themselves, or properties thereof in hierarchical NNs (HNNs)\cite{30, 28}. These nets use two distinct encoders, one for the utterances in the context, the other for the encoded utterances. So dialogue contributions are processed by a standard encoder (the language level), generating encoded representations which in turn are processed by another sequence coder, now working on the dialogue level.

1.1 Hypotheses

Although the aforementioned dialogue system showed that content can be learned to some extent by the seq2seq model, a problem could arise when the system should learn specific information which becomes available in the course of the conversation. This is especially important if the goal is a widely applicable dialogue agent. Since the seq2seq model has some memory build in such that information can be kept over time, it performs well on sequences, in which long term dependencies are important. The seq2seq model showed it is very capable of language modeling, with most significant and illustrative results in machine translation \cite{6}. In this work I will test external memory modules and hierarchical architectures. I hypothesize the internal memory to be responsible for language modeling and external memory for content tracking (h1).

Getting back to previous observation, language modeling is closely related to the superficial dialogue task, but it seems to require some world knowledge. For example, when ‘red’ is the previous word, it is highly likely followed by a noun referring a concrete object rather than an abstract phenomenon. In dialogue modeling, this world knowledge is considered to be part of what is assumed to be known to dialogue participants by all other participants and referred to as common ground. When a conversation proceeds, more common ground is created (additional to world knowledge), since contributions inherently contain information. This newly created common ground is related to the more difficult task of content creation and I hypothesize that external memory modules will be able to keep track of this information more successful than the plain seq2seq (h1).

Furthermore, content reasoning can benefit from information about what kind of contribution to the ongoing dialogue is required \footnote{In the case of human contribution the verb ‘desired’ or ‘intended’ is more appropriate, but in this case an artificial system is subject.}. In dialogue modeling, contributions can be classified by dialogue acts (DA’s) \cite{32}, which do exactly this. To include this
information in generation of a reply, the desired dialogue act is needed before generating a contribution. This information can be made available to the system two distinct manners. Firstly, the most straightforward way, the DA of the target is also given to the decoder. Secondly, I hypothesize that the target is one of many optional contributions and therefore other DA’s would be acceptable as well (h2). So by using the DA’s of the n sized context (of contributions), as well as the textual information, the system should be able to infer a DA implicitly itself. In this case, I hypothesize that external memory will not make a significant difference since the memory of the seq2seq is now not required to capture dependencies in the language modeling (h4). But as shown in [28], I expect the hierarchical setup to outperform a standard seq2seq (h5).

Finally, information is accumulated during progress of dialogue, a range of context sizes is used for input in the experiments. Since external memory has proven to be more powerful on reasoning, question answering and planning tasks [10, 22] I expect that the external memory modules will outperform the basic seq2seq with growing context sizes (h6).

1.2 Overview

In the following section the theoretical framework is outlined in which this work is placed. Starting with dialogue properties in section 2.1 to explain the different kinds of information involved. This is followed by a description of the probabilistic models for language (and dialogue) in section 2.2 their technical properties whence implemented in RNNs are described in 2.3. With these basic building blocks defined, in section 2.4 these are constructed in a hierarchical setting. External memory is a module which can be fitted on different parts of the (H)RNNs and are described separately in section 3. Not all possible combinations are tested, an overview of the tested models is given in section 4. Then the method and experiments are described in sections 5.1 and 5.2 respectively. Direct results, i.e. graphs of individual model performance, are also given in section 5.2. The results are further discussed in section 6.

Not all hypotheses can be confirmed and results are worse than is to be expected, also for the baseline models. Although, since the models do show some consistent performance, it can be concluded that the NSE is able to modify content, and that the hierarchical setting should be preferred over the seq2seq in further work.
2 Neural Models for Dialogue

To understand models for dialogue, the statistics of dialogue (and language) need to be understood, and before that dialogue (not language) in its most abstract form. This section starts out on theory of dialogue, followed by statistical models of language, from which together probabilistic models of dialogue can be given.

In order to understand the modularity on the technical side, this chapter concludes with a framework in which state-of-the-art NNs architectures are described. Additionally this chapter contains a description of NN architectures which are exceptionally suited to test the modularity on the theoretical side.

2.1 Dialogue

Much work has been done on dialogue modeling on which can be drawn when looking for inspiration for AI model design. A comprehensive overview of dialogue modeling is given in [7] of which here relevant concepts are summarized.

In dialogue modeling we cannot use the term ‘sentences’ formally, since often it is not clear when a spoken ‘sentence’ should begin or end. Therefor it is much harder to use grammatical based models such as context free grammars and parse trees. Especially in spoken dialogue, the notion of a grammatical sentence is of little help. Speakers interrupt others to complete what is being said, specify a misunderstanding or ask for clarification. All of which are pieces of information to support the dialog.

To denote a sequence of words in dialogue, researchers speak about turns and utterances, which can be composed of any number of the other.

\[\ldots\text{informally turns may be described as stretches of speech by one speaker bounded by that speaker’s silence – that is, bounded either by a pause in the dialogue or by speech by someone else.}\]

whereas

\[\ldots\text{an utterance may be described as a unit of speech delimited by prosodic boundaries (such as boundary tones or pauses) that forms an intentional unit, that is, one which can be analyzed as an action performed with the intention of achieving something.}\]

This description of an utterance points to an important aspect of language use in dialogue: the literal content is accompanied by (non-literal) information which can modify the meaning of the former. For example, consider the sentence: The cat sits on the table. When this is spoken and brought as a mere fact it can function as an answer to the question of where the cat is. But when spoken with amazement, the message changes and the receiver of such an utterance also becomes aware of the fact the speaker is surprised about the fact that the cat is sitting on the table.

\[\text{Since this is assumed to be common ground because we maybe even think with it.}\]
2.1.1 Dialogue Acts

So when something is said, apart from the information composed of the meaning of the words, an intention is also part of the message. This was described in the past century in [11] and in this respect an utterance can be seen as an action (most definitions of intention make use of bringing something about in the world). This notion gave rise to speech act theory [3], which describes levels on which actions operate for a particular utterance. What is being said, the locutionary act, what the intention of the speaker was when uttering the utterance, the illocutionary act and what effect it has on the hearer, the perlocutionary act. All three acts are always present in an utterance, but only the first two are used in digital applications. The locutionary act does not differ from meaning in other language settings (such as written text) and is therefore straightforward to implement using NLP techniques. But the illocutionary act is important to utterance generation. The notion of acts is extended and developed over the years (see [27] for example), and put to a formalization of dialogue acts in [1]. This schema of dialogue acts has been extended and modified to fit data-sets in order to have a classification scheme to extract or use the extra information in utterances.

Classification of dialogue acts was a popular topic of research in AI with the emergence of the seq2seq, and state-of-the-art models score above 75% precision [16]. This is a relatively low score in general machine learning problems, but in the case of dialogue act classification the distribution of utterances is extremely skewed.

Nevertheless, the sequential distribution of classes over utterances in a conversation is not and therefore predictable [28]. It has proven to be beneficial to use the dialogue acts of preceding utterances as features in predicting the dialogue act of a given utterance.

Now, given the fact that dialogue acts can be predicted with reasonable accuracy, I will test whether information on dialogue acts of previous utterances can increase performance of utterance generation, as is already shown in [30] for two previous utterances.

2.1.2 Grounding

With the perspective of utterances as actions, dialogue can be studied as a joint action model [7]. This approach tells us that participants of a conversation are cooperating to achieve a common goal, i.e., to increase the amount of information known by all participants. Obviously, before starting a conversation people assume a great deal of knowledge to be known by the other participants.

An important aspect emerging from the joint action model is that of grounding and alignment. When we speak with each other we assume certain things known to the other party in advance, like what phenomenon is named ‘weather’ and what the weather was yesterday. This is what is called common ground [31]. Formally it is the intersection of all things that are known to the participants of the conversation. Now, as the dialogue progresses, more shared knowledge is added to this set. But the process of adding facts sometimes needs some coordination and utterances with different acts are used. This process is called grounding. The core aspect of grounding is to have common knowledge about what a word means, therefore people tend to use the same words for the thing which is subject of the conversation [8].
This phenomenon is observable in repeated use of the same word over a range of utterances. For example in a booking situation, when a particular city is named in the context multiple times, it is desirable the system will use it as well in reply generation, although the language model would prefer another city in this context.

2.2 Probabilistic Models

As is described in the previous section, there is a lot more to dialogue than the literal meaning of a sequence of utterances. A particular utterance has multiple layers of determinants that together compose the actual meaning. At the base lies grammaticality, which is commonly referred to as the language model. But from the space of all possible grammatically correct utterances, only a tiny part is acceptable as a natural contribution to a particular conversation. The acceptable space is hypothesized to be restricted by all preceding utterances up till that point in the particular conversation.

In the following explanation the set of utterances which are hypothesized to be determinants of acceptability of a to be generated utterance is referred to as context.

2.2.1 Utterance Model

An utterance can be given a probability of being an acceptable contribution to the conversation. As explained above, it needs to be grammatical. In NLP this is nowadays achieved by use of a Recurrent Language Model [20]. The model is recurrent due to the nature of its implementation and not of importance here, but will become clear in section 2.3.4. It defines a distribution over a sequence of words by the joint probability, which in turn is defined by the product of the conditional probabilities of word \( i \) following the preceding ones:

\[
P(\text{utt}) = P(w_1, w_2, \ldots, w_N) = \prod_{n=1}^{N} P(w_n|w_1, w_2, \ldots, w_{n-1})
\]

2.2.2 Dialogue Model

Technically, a dialogue is an equivalent sequence, but the elements are utterances instead of words. So we get a distribution over dialogues as follows:

\[
P(\text{dialog}) = P(\text{utt}_1, \text{utt}_2, \ldots, \text{utt}_M) = \prod_{m=1}^{M} P(\text{utt}_i|\text{utt}_1, \text{utt}_2, \ldots, \text{utt}_{m-1})
\]

But the basic input features in NNs are still words and we would need to calculate the following for every word (which is infeasible).

\[
\prod_{m=1}^{M} \prod_{n=1}^{N} P(w_{m,n}|w_{m,<n}, \text{utt}_{<m})
\]

\(^{10}\) Although work is done on character input as well, state-of-the-art models use words.
In the successful dialogue system mentioned in the introduction, this is done with only the last preceding utterance, i.e. to generate utterance \( i \), utterance \( i - 1 \) is context. In other works a context of two utterances is used, but to my knowledge never much more to reach state-of-the-art results.

In this work I will increase the context size and investigate whether hierarchical modules can be used for the different levels in dialogue. In order to do so, the grammaticality is modeled to be directly dependent on the last previous utterance (as is the case in [37]), but can be modified with information from context. Such models can be described by the following distribution:

\[
P(utt_i) = P(init\_reply, utt_{i-1}, utt_{i-2}, \ldots, utt_{i-m}) \times P(init\_reply|utt_{i-1})
\]  

(4)

where the \( init\_reply \) represents an initial string of words generated by a language model. The hypothesis is that given this grammatical string, another module, in the next processing step can adjust words or phrases to account for grounding and coordination or style in the case of DA context. Three different modules are tested on a varying context size, which will be explained in section 3.

2.3 Generative Neural Networks for Language

The emergence of NNs in NLP is mainly result of the insight to represent words as vectors, where the numeric elements represent a semantic distribution in terms of other words[21]. Furthermore, a NN which can handle arbitrary length input is needed to process sentences or utterances. To this end recursive NNs are developed. The combination of these two techniques enable computers to generate new strings, and thus contribute to dialogue.

2.3.1 Semantic Embedding

‘The meaning of a word is defined by the company it keeps.’ A phenomenon first noted by J.R. Firth in 1957 and a famous statement in the NLP community, not the least since it so appealing to common sense. In [21] an algorithm is presented which construct vectors that encapsulate exactly this idea. Several distinct manners exist nowadays to construct such vectors, but a NN NLP model can also learn the vectors simultaneously with the model and thus capturing only the semantics of the words in the training-set.

In practice, when modeling language, one uses the set of words occurring in the training-set, kept in fixed order in a vocabulary. Input to the neural net is a one-hot vector indicating the index of the word in the vocabulary. But since the number of words is often very large (> 50000), an embedding matrix is used to reduce the dimensions. This embedding is a matrix of size \( v \times d \) (where \( v \) is the number of words in vocabulary and \( d \)

\[\text{See this Wikipedia entry.}\]
the desired dimension of wordvectors) and represents the weights of embedding. When
inputting a word (one-hot vector) one of these rows is selected.

The weights can be learned simultaneously with the model or separately on a much
larger dataset than the actual training set. By pre-training wordvectors, broader seman-
tics can be captured than when ad-hoc trained, which capture only semantics present
in the training-set. Several different training objectives are used nowadays to obtain
pre-trained vectors. In the *skip-gram* method the objective is set such that the word
representation predicts the context words. And vice versa, the *cbow* method results in
representations which predict the represented word given the context. So a set of these
vectors combined results in the representation of the word which is most often found
in this specific context. *Skip-gram* and *cbow* are introduced in [21] and more specific
in [19]. Both methods have their advantages, but in the seq2seq framework both are
desired. I.e., when encoding a sentence the *skip-gram* is more desirable, in generation
*cbow*. Therefor I will use a third pre-trained vector type in this work, known as *GLoVe*,
which combines the advantages [24].

When the model outputs a wordvector (models perspective) it is transformed by a linear
layer to the dimensionality of the vocabulary, but is generally not one-hot. In the NLP
community the LogSoftMax (natural logarithm of equation [5]) trick is used to get a
probability vector over the vocabulary. This function is differentiable and can thus be
used to define a loss function required for training the net with backpropagation.

\[
\text{Softmax}(v_m) = \frac{\exp(v_m)}{\sum_{i=1}^{M} \exp(v_i)}
\]

These distributions over the vocabulary per outputted word allow, in combination with
use of equations [1] and [2], evaluation of the loss of an utterance which is commonly
referred to as perplexity, and thus train a NN to generate utterances. The loss is
determined by the difference with respect to the target utterance in the training set,
a 0-loss is achieved when all words in the target sentence are generated by the model
with probability 1. Perplexity is defined by

\[
2^{-MLE}
\]

where *MLE* is defined by the non negative of equation [6]

\[
\text{Loss(utt)} = -\sum_{i=1}^{n} \log P(w_n|w_1, w_2, \ldots, w_{n-1})
\]

\[
P(w_n = v_m|w_1, w_2, \ldots, w_{n-1}) = \text{Softmax}(v_m)
\]

Actually, this is a poor measure for dialogue and language modeling in general, since
other utterances could be scored as high or higher by human judges. For example, if
a system should reply on: ‘What do you think of [some city]?’ and in the training-set
city names are only encountered in context with aesthetic valuations on architecture,
the system would probably respond with this kind of valuation. But a reply referring
to the atmosphere or possible cultural activities would be as good, or better, depending on context.

Perplexity is used nowadays because there is no good alternative yet. Currently much work is done in reinforcement learning, where the naturalness (or inverted loss) is given by another NN, but at time of writing it still has not been proven more effective than perplexity.

2.3.2 Recurrent NNs

Dealing with language means varying input sizes. If we let an utterance be a sequence of words \(\{w_1, w_2, \ldots, w_i\}\) it follows naturally to input the words in the order they appear in. Then the dependency between words (as defined in equation 1) is modeled by a NN which maps \(w_i \rightarrow o_i\). At processing the next word, \(w_{i+1}\) and \(o_i\) are both used as input. These type of networks are coined recurrent NNs (RNNs).

In NLP, these RNNs are nearly always equipped with internal memory to account for long term dependencies in sequences. The most well-known memory is the Long Short Term Memory (LSTM) [12]. The memory is implemented in each node of the network and is therefor internal. The LSTM was proposed as solution to the ’vanishing gradient’ problem, and boosts performance of sequence processing due to word dependencies spanning larger parts of utterances. These properties are very much related, but the vanishing gradient is more simple, i.e., it is a technical quality which is solved by the LSTM. The fact that long term dependencies are modeled more successfully is result of the solution to the vanishing gradient. All in all, it has proven itself in many NLP applications.

2.3.3 Long Short Term Memory

As mentioned above, an LSTM solves long term dependencies between words in sequences, but the range has limits. In order to understand how it deals with these, in appendix A.1 the equations that govern the forward behavior are listed, accompanied by a schematic presentation. The LSTM node has the capacity to store information in its ’cell state’ which can be used to compute output of later steps by keeping in memory (the second part of equation 31 is small and the first bigger). This kind of memory is dependent on the input, and, since it is a recurrent layer, on all preceding inputs. The nodes are typically used in layers of several 100 adjacent nodes, in one till three layers. A clear result of this short term memory is the ability to be quite capable of simulating language models as is shown in [20]. For example, the cell states can store gender type of a subject, and release the information at the point of selecting the correct pronoun. But when the context size grows, more dependencies exist, and one would need more nodes to capture all of them. A larger number of nodes implies more parameters and thus the need of more data. To overcome this shortcoming, this work experiments with external memory to alleviate the cell states of dependencies on dialogue level and let them do their work on language modeling level.
2.3.4 Sequence-to-Sequence Model

With the language model taken care of, in dialogue we need to be able to model the dependency on the previous utterance. In [33] an architecture is introduced to do exactly this. Although the model is presented on the task of Machine Translation, it is also suitable for dialogue. This model consists of an encoder, and an output generator, or decoder. In principle the decoder can adopt any architecture, depending on the required output, but in the case of dialogue is should generate an utterance and is therefore also an RNN, and hence the name ‘sequence-to-sequence’.

At run-time, an input utterance is processed by the encoder which yields a final output. This output, together with the memory cell states represent the encoded input utterance and is used as initial state of the decoder. In training time the decoder is generally also fed with the target for better generalization. See figure 1 for a schematic of this famous system.

The last output of the model encoder ideally represents the entire encoded input sequence. This is in literature referred to as the hidden state and turned out to be not enough in practice. The extension of attention mechanism (explained in section 3.1) has made it one of the most popular models in NLP.

2.4 Hierarchical Recurrent Neural Nets

Since the focus of this work lies on the different levels of dialog, hierarchical NNs (HNN) are a suited tool to investigate this. In [28] it is shown how context can be used in a hierarchical setup to improve replies in dialogue. In this work a dataset named MovieTRiples is used. This dataset consists of triples from movie scripts. All triples start with speaker $A$, then an utterance of speaker $B$ followed by the target utterance of speaker $A$ again.

The system encodes both context utterances with the same RNN, this yields two vectors representing the context, these are then processed by another RNN, the session RNN, which’ output serves as input for the decoder to generates the target. See figure 2 for schematic presentation. The system was originally presented in [29] to predict the next search query in a user session on a search engine.
This setup explicitly models a dialogue state which is updated on every utterance. The session RNN’s encoded state is used as additional input on generation of each token in decoding and in this way can incorporate the information other than a regular recursive language model. When using the text of context utterances it remains unclear what role different kinds of information within the context add to performance.

Another approach, presented in [35], models the dialogue act sequences explicitly. A reply is set to be dependent on the expected dialogue act as well as the previous utterance. The system simultaneously predicts $P(da_i|utt_{i-1}, da_{i-1})$ and $P(utt_i|utt_{i-1}, da_i)$. In figure 3 the system is depicted. Context (the last utterance) is encoded using an RNN and dialogue act classification is done with a custom prediction layer (see the original paper for the details).

These architectures will in this work be combined and enhanced with external memory modules, which are described promptly.
3 External Memory

As mentioned in section 2.3.4, I hypothesize style and alignment could better be modeled with other modules than the standard seq2seq, especially because this information is originating from a larger context than the last preceding utterance. For this purpose I will investigate ‘external memory’, which is an extension on the attention mechanism.

In this section the attention mechanism is explained in detail, followed by the two different external memory setups used to test the hypotheses.

3.1 Attention Mechanism

Nowadays, seq2seq models are often extended with the attention mechanism [4]. This extension makes encoder states available to the decoder as last processing step when generating a word. The encoder outputs are generally referred to as context (not to be confused with context in dialogue, but the meaning of context being encoder outputs is only applicable in this section). It has been proven a highly effective extension in many, if not all, NLP tasks. A great benefit is the low number of extra parameters, and since the context is a leaf node in the computational graph not much extra training resources are required. Attention increases performance because at every decoding time-step all encoder information is available, opposed to having only information available via the encoder hidden state.

The attention mechanism introduced in [4] is improved over the years, but the intuition is as follows. Let \( H \) denote the set of all \( h_i \), i.e. encoder intermediate output states and \( s_t \) decoder output at time step \( t \), then key is yielding a probability vector representing importance of each \( h_i \in H \) for this particular \( s_t \) yielded by the decoder. This ‘attention’ vector represents the weights in a summation of \( H \) which is then combined with \( s_t \) to form the final decoder output.

In this work the following definition is used, as is in [13], with two linear layers: \( W_{in} \) and \( W_{out} \)

\[
\begin{align*}
\text{att}(H, s_t) &= s_t^* \\
q &= W_{in} s_t \\
\text{sim} &= \text{softmax}(qH) \\
a &= \text{sim} H \\
s_t^* &= [a, q]W_2
\end{align*}
\]

where \([\cdot, \cdot]\) is concatenation.

Essentially this mechanism can be seen as external memory since the encoder states are explicitly saved for later input. And no problem arises when using more previous utterances as context (both meanings apply here). Furthermore, alternatives for composition of \( H \) can be used, but the memory is always leaf-node, external memory changes exactly this aspect.
3.2 Neural Semantic Encoders

3.2.1 Semantic Encoding

In [22] the Neural Semantic Encoder (NSE) is introduced and uses the attention mechanism as explained above. Although the equations are very similar, the manner in which it is used is very different, i.e. it modifies the context and hence it is called external.

For input pairs $X_i, Y_i$ with $x_1, x_2, \ldots, x_k \in X_i$ as embedded words, the memory is initially filled with the embeddings (of size $l$) of the single input sequence (of length $k$) and is updated at each encoding iteration. So memory matrix $M$ has size $k \times l$.

The system consists of a read, compose and write module and is depicted in figure 4. The read module is an LSTM which takes input $x_t$ and produces output $o_t$, which is compared to each memory cell individually. The output of this comparison yields a vector of size $k$ (sequence length) and is softmaxed to represent a probability distribution over most similar memory cells. Note that this is exactly the same as the standard attention mechanism as described above. This probability vector is coined $z_t$, or location vector. Now the product of this vector and the memory (matrix) yield a ‘read vector’ which is combined with $o_t$ (the output of the read module) by the compose module, a multi layer perceptron (MPL). It’s output $c_t$ is input to the write module, which is again an LSTM. Finally this output, $h_t$, is written to memory at location $z_t$. Note that $z_t$ is softmaxed and thus often not one-hot. The write operation first erases the read information before writing.

Formally:

$$o_t = f_r^{LSTM}(x_t) \quad (13)$$

$$z_t = \text{softmax}(o_t^T M_{t-1}) \quad (14)$$

$$m_{r,t} = z_t^T M_{t-1} \quad (15)$$

$$c_t = f_c^{MLP}(o_t, m_{r,t}) \quad (16)$$

$$h_t = f_w^{LSTM}(c_t) \quad (17)$$

$$M_t = M_{t-1}(1_{l \times k} - z_t \otimes 1_{k \times 1})^T + (h_t \otimes 1_{l \times 1})(z_t \otimes 1_{k \times 1})^T \quad (18)$$

The NSE has shown to beat state-of-the-art approaches on inference \textsuperscript{12}, question answering \textsuperscript{13}, sentiment analysis and machine translation.

In many NLP application input consists of multiple sequences of words, the NSE can be extended to a Multi Memory Access NSE in the following manner. Instead of a single, we get $n$ memory modules and $n$ replications of equations (14), (15) and (18). In equation (16) all memory slots are then used as input for the compose module. (See appendix A.2 for a full listing of equations.)

\textsuperscript{12}Stanford Natural Language Inference. \textsuperscript{5}

\textsuperscript{13}Selecting an answer from multiple options. \textsuperscript{38}
The memory of NSE can be directly compared to the encoder-context. In this setup though, the encoder itself is optimized on the resulting encoded states of the sentence, as well as on the encoded memory sentence. Because of this split computational paths, the reasoning NSE uses a disconnected encoder-decoder structure, and thus thereby optimizing the encoder on reading of (or attending over) input only.

### 3.2.2 Reasoning NSE

In the hierarchical setup, the layer on which the NSE will work is responsible for dialog level information. Therefore the following version of the NSE is tested.

Quickly after publication of the previously mentioned paper, the same authors use the NSE to tackle hypothesis testing [23]. Given a document, a query and a set of possible answers, a different version of MMA-NSE is used to evolve the query to a hypothesis containing the correct answer. Two memory modules are used, one for the document, one for the query, both implemented by a bi-directional LSTM, i.e. $M^x = bi-LSTM(X)$. The document memory is only read, the query memory state also updated. The system is defined by the following equations:

\begin{align}
    r_t &= LST^M([s^q_{t-1}; s^d_{t-1}]) \\
    l^q_t &= r_t^\top M^q_{t-1} \\
    s^q_t &= \text{softmax}(l^q_t) \top M^q_{t-1} \\
    z^q_t &= \text{sigmoid}(l^q_t) \\
    l^d_t &= s^q_t \top M^d \\
    s^d_t &= \text{softmax}(l^d_t) \top M^d \\
    c_t &= MLP(s^q_t, s^d_t, r_t) \\
    M^q_t &= M^q_{t-1} z^q_t + s^d_t (1 - z^q_t)
\end{align}

With this setup this reasoning NSE must be equipped with a halting mechanism to stop updating. In [23] two methods are presented, of which I used the following. A hyperparameter is set to ensure a maximal number of writes. But the system itself can also learn when to stop (before the maximum is reached). This becomes apparent in equation 26, when $z^q_t$ from eq. 22 is approaching 1 on every element, i.e., the similarity in eq. 20 low, no new information is written, and the query memory is hardly altered. This is named the gating mechanism.

Note that the seq2seq architecture is not present anymore and information flows only through application of the attention mechanism.

### 3.3 Differentiable Neural Computer

A wholly different kind of external memory is presented in [10] and an extended version of the model in [9]. There are two main components: the controller (comparable to a CPU, but differentiable) and a memory matrix (comparable to ram).
The controller has an LSTM architecture to process sequences of input. At each time step it emits an output vector $u_t$ and an interface vector $\xi_t$, which regulates interaction with memory at $t$. The input is composed of data input, $x_t$, concatenated with a read vector from memory, $r_{t-1}$. The interface vector is a composition of read keys and read strengths, write keys and strengths, an erase vector, a write vector, free gates, the allocation gate, the write gate and read modes. So at each time-step the system can write to memory, and read information to input along with the next data item in a sophisticated manner.

Furthermore, it is equipped with a memory usage vector to keep track of usage per memory row, and on top of that a temporal link matrix to keep track of sequentiality. See figure 5 for a schematic overview.

The system is capable of content based addressing for reading by combining the read keys and strengths to yield a read vector. Writing to memory is dynamic and regulated by a memory usage vector, which, in combination with the free gates from the interface vector, determine how much can be written to a location. To keep track of the order of accessed locations a temporal link matrix is used, which can shift a weighting (addressing) forward or backward.

**Difference from NSE**

Clearly this setup is much more sophisticated than the NSE. It is designed to reason, whereas the NSE works on semantics of words. Furthermore, the DNC is designed to reason and solve puzzles, so it is hypothesized in this work to improve performance when given dialogue acts.
4 Models

The preceding sections give rise to many possible combination of model setups. But keeping the hypotheses in mind, only some are directly compared in experiments. The models on which experiments are done are of the following types.

4.1 Baseline Models

The baseline model is a standard seq2seq with attention as described in section 2.3.4, this model will be referred to as baseline. Since the DNC has identical input/output as the sequence coders used in the baseline model, it can replace them, yielding another model referred to as dnc-dnc, in which the attention mechanism is replaced by the DNC memory which is shared between the en- and decoder. These two models will serve as bench-marking the DNC on context processing while also being responsible for language modeling.

4.2 HRED Models

The HRED is described in section 2.4 and depicted in figure 2. This model is implemented as described there and extended with the attention mechanism over the dialog states. This model will be referred to as the base-hred. A next model is result of replacing the dialog level encoder- and decoder-LSTM by a DNC, coined the lstm-dnc, as is the case with the dnc-dnc, the attention mechanism is dropped here.

The base-hred model is also tested with DA’s. To test whether DA’s can increase performance, the DA of the target utterance is embedded and concatenated to the final dialog encoder hidden state and thus input to the decoder. Since, theoretically, the possible target DA could be different from the one given, this model is also tested by using the context DA’s used as input and without making use of the target DA. In this case the dialog level encoder receives a vector which is result of concatenation of the utterance hidden state with utterance DA, put through a merge layer. This way the model can learn to either use or not use DA information from context.

4.3 Custom Hierarchical Models

Since the Reasoning-NSE is inherently hierarchical, its method of attending over input is compared to others, by keeping the Reasoning-NSE architecture and replacing only the attention mechanism (or external memory variant).

First, the attention mechanism is replaced by the standard, effectively breaking the backward computational path. Second, the DNC is modified to work exactly as the Reasoning-NSE, effectively changing the type of external memory.

This setup allows for a comparison of modules under the assumption that language- and content modeling can be unraveled technically as well as theoretically. The LM component uses only the last utterance to generate a initial reply (in figure) from. See figure for a schematic. This init_reply is then modified by the aforementioned 3 variants and on different kinds of information in context, i.e. DA’s, keywords and text.
Let any type of \( n \) context utterances be denoted by: \( C = [c_1, c_2, \ldots, c_n] \), then the basic attention and DNC use each piece separately as shown in figure 6.

**Basic attention**
The baseline is set using a bi-LSTM of 300 \( \times 2 \) nodes to provide the encoded context vectors \( \text{enc.c} \). At encoding each \( c_i \), the hidden state of the encoder \( (\text{enc}) \) is reset to 0. The initial reply is then inputted to the basic attention model which attends over the set \( \text{enc.c}_1, \text{enc.c}_2, \ldots, \text{enc.c}_n \). So in figure 7 the \( \text{encs} \) are implemented with LSTMs and the decoder as only an attention layer.

**DNC**
To compare the DNC with the basic attention scheme, the bi-LSTM and attention layer from previous model are replaced by a DNC (each \( \text{enc} \) and the decoder in figure 7 now represents a DNC which share their memory). Both controllers are implemented by an LSTM layer of size 300 \( \times 1 \). The memory consists of 40 cells, each containing 100 entries in which information can be stored.

It processes each \( c_i \in C \) as a separate sequence, meaning the hidden states are initialized at 0 for each piece of context. At decoding, the layer receives the init_reply sequentially (with again a fresh hidden state) and uses the memory to update the initial words to possibly new ones.

### 4.4 Reasoning-NSE

The Reasoning-NSE was introduced as a algorithm to transform a question to an answer in [23]. Since the regular NSE as described in section 3.2.1 performed worse than the Reasoning-NSE in preliminary tests, only the last has been used in the experiments and will be referred to as nse.

The Reasoning-NSE does not share the input/output specification of the LSTM and is therefore tested in a semi-hierarchical setting like the ones described above. Contrary to the other models, the tested NSE concatenates all \( c_i \in C \) which serve as the document memory, the init_reply serves as utterance memory. In figure 8 the setup is depicted. The document memory is shown as context memory, as it will be referred to in the rest of the paper. As explained in section 3.2.2 the context memory is initialized by a bi-LSTM. The utterance memory is not, since it is provided with the init_reply which is results of a LSTM of which the last hidden state serves as initial state for the decoding NSE.

The read and write modules are implemented by single layer LSTMs of also size 300. In all experiments maximal five reasoning steps are completed to alter the utterance.
memory. After these steps the utterance memory represents the final reply.

Additionally, the reasoning NSE is tested in a similar way as the base-hred. The target DA is embedded and included in the concatenation just before the compose step (just before equation 25).
5 Experiments

The models are implemented in Python and run in PyTorch. The code of the models, train- and test routines are based on the OpenNMT system. This section gives a description of the methods used. Starting out with a specification of data, training method and evaluation metrics. Note that GloVe word vectors are used which are in some cases included in training. The last part of this section contains a description of the specifics of each experiment including results.

5.1 Method

5.1.1 Data

Switchboard dataset

The Switchboard dialog act dataset is a set of telephone conversations on randomly assigned topics of two participants who do not know each other beforehand. The set consists of 1155 conversations of totaling 198000 utterances. Since the set represents spoken language it contains many occurrences of ‘uh-huh’ and ‘uh’, these are removed. The dictionary has 19413 words, of which 1215 are unknown in the GLoVe word vectors. Context sizes ranging from 1 to 9 are extracted. If an utterance has length less than 2 or more than 40 tokens it is discarded. The context consists of consecutive utterances, so when an utterance is discarded or a new dialog starts, a new context is created. The numbers of data points in training and test are listed in table 2. The sets also contain the dialog acts, the original 210 dialogue acts are used, so no clustering is applied in building the data sets, this is left to the models to work out during training.

<table>
<thead>
<tr>
<th>context size</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>training</td>
<td>151612</td>
<td>134834</td>
<td>120250</td>
<td>107544</td>
<td>96477</td>
<td>86794</td>
<td>78287</td>
<td>70814</td>
<td>64214</td>
</tr>
<tr>
<td>test set</td>
<td>19256</td>
<td>17129</td>
<td>15300</td>
<td>13725</td>
<td>12331</td>
<td>11109</td>
<td>10040</td>
<td>9107</td>
<td>8283</td>
</tr>
</tbody>
</table>

Table 1: Data points in data sets.

Frames dataset

The frames dataset is a collection of dialogues from hotel reservation system, created for development of dialogue systems. The task in creation of the dataset is for a user to book a room by conversing with the wizard, which has access to a database. The set contains 19986 turns spread over 1369 dialogues. For all experiments the dataset is split in a fixed 90% for train set and the remaining 10% for testing. The dictionary size is 3852, when using pre trained word vecs, 160 are unknown, mostly typo words and symbol strings.

The dataset is presented in [2]. Dialogues are annotated with dialogue acts and frames. There are 20 dialogue acts, under which some dataset specific ones like switch, frame,
request, compare and request alts, the rest are similar to the dialogue acts described in section 2.1.1. A frame represents a certain option for booking, a new one introduced when the user changes something in the request or the agent offers or suggests another option than is discussed in the recent frame. A frame is technically a set of keywords about the frame, or relation to others, and named entities and values, such as locations and prices. All are handled identical, a separate vocabulary is constructed and can this way represent an utterance in the context.

<table>
<thead>
<tr>
<th>context size</th>
<th>1</th>
<th>3</th>
<th>5</th>
<th>7</th>
<th>9</th>
<th>11</th>
</tr>
</thead>
<tbody>
<tr>
<td>training</td>
<td>7381</td>
<td>6203</td>
<td>5068</td>
<td>4018</td>
<td>3018</td>
<td>2276</td>
</tr>
<tr>
<td>test set</td>
<td>831</td>
<td>698</td>
<td>570</td>
<td>450</td>
<td>344</td>
<td>257</td>
</tr>
</tbody>
</table>

Table 2: Data points in data sets.

5.1.2 Training

All models are optimized on data sets different from those described above and thus different from those which are used to report results on. The data sets on which optimization was done include Machine Translation sets and Open-Subtitles.

The models are trained with Adagrad and conditioned on cross entropy loss. The learning rate is set on all experiments to 0.0001, decay starts at epoch 9 if validation loss does not increase earlier and decay has a value of 0.75.

Dropout is implemented after each LSTM layer, the DNC profited most of a probability of 0.6, the regular LSTM layers with 0.4 and dropout on the NSE was set to 0.2. All models used word vectors embedded in 300 dimensions, in the results section it will be reported when pretrained were used. All models were trained till convergence, i.e. until the validation loss started to increase.

5.1.3 Evaluation

In this work the models are trained on perplexity, which is, as mentioned in section 2.3.1, a poor measure, but has also proven itself to be able to produce models with language modeling capabilities. Therefore, assuming language modeling does not suffer from more relevant information, a more stable and possibly increasing score is expected when tuning up the context size.

A word-count known as ‘term frequency-inverse document frequency’ (tf-idf), introduced in dialogue system assessment in [17], is a possible measure to test whether memory models perform better at coordination and alignment. But since both, perplexity and td-idf, are not as good as human judges [15], the results are subjected to a qualitative analysis. Since it is infeasible to check all generated replies a subset of these is analyzed based on the score of the reply given the model, i.e. only the replies with a high probability are subjected to a qualitative analysis. It is important to note that comparison of the models will be based on their relative performance over the varying context size.
5.2 Experiments & Results

This section gives a description of the experiments, accompanied by the results, displayed in graphs showing model performance over an increasing context size. The numbers in the legends indicate the number of parameters. Furthermore, in the example tables, source indicates the last preceding utterance which is used to get an initial reply from.

5.2.1 Key Word Context

To test whether already important information of the utterances in context can be used more effectively with external memory than the standard attention mechanism, the hierarchical hred-base, lstm-dnc and nse are compared. The word vectors are pretrained and updated during training to suit the dataset.

The key word context experiments are run on the Frames dataset only. Figure 9 shows the results of experiments on a varying context size consisting of keywords. The nse continuously performs better than the baseline and the lstm-dnc. Furthermore, the DNC shows no performance gain over the baseline, which is unexpected.

![Figure 9: Context in key words.](image)

Although, a closer inspection of the replies on the testset, the nse seems to distort the initial replies. See for example table 3 from the test with 3 utterance in context. great! has been changed in alright beans which doesn’t make sense, but trip is changed to flight, which is more specific.
### Dialogue Acts

The other important information type, the DA’s, is tested in two settings. Firstly, with the target DA specified to the decoder. In which the dnc-dnc is compared to attention. The results are plotted in figure 10. A qualitative analysis of the replies showed no promising results and it seems like the DA information only confuses the model into ungrammatical utterances.

![Figure 10: Target dialogue act on frames dataset.](image)

The second experiment on DA’s was done in a hierarchical setting as is done in [28], the base-hred with target DA (in the figure: da-baseline). And to test the hypothesis that the target DA is just one of the possibles to generate an appropriate reply with, its also tested with DA context, joined at the dialog-level encoder in a hierarchical setup, in the figure: lstm-hierda. The results are plotted in figure 11. The minute difference in performance of the hierarchical and non-hierarchical baselines possibly show the DA information is either equivalent or irrelevant. The nse in this case receives the target DA and uses textual context.
The Reasoning-NSE shows much better performance than the other models in the experiments on the Switchboard dataset. But when analyzing generated replies this does not show consistently. These results showed the most ungrammatical replies, with much repetition of words.

5.2.3 Text Context

Both uses of explicit information as context did not produce good results. But since this information should be included in the textual utterances, the models are less constricted to infer information from the textual representation of the utterances.

Firstly the nse is compared to the baseline. The results are plotted in figure 12. Note that this is run on the Frames dataset and hence the irregularity. The improvement of the nse over the baseline is most probably an artifact of the number of parameters, since no significant difference in quality of replies is found.

In this case the nse seems to distort the initial replies as well, but the examples from tables 4 and 5 both show some interesting properties. In table 4 we see that sure is inserted while not changing much of the meaning by leaving out would. In table 5 we see again a change by inserting a confirmation at the start of the reply.

Finally, textual context is also tested in a hierarchical setting. In these experiments the DNC and baseline are both used in hierarchical and non-hierarchical form. Results are found in figure 13. Apart from the nse, no models outperformed the baseline, which is the ‘good-old-fashioned’ encoder-decoder.
Figure 12: Text context on frames dataset.

Table 4: Example from NSE reply on text context.

| source                                      | ‘ is that for me and my friend |
| context                                     | we ‘ll be leaving from vancouver |
|                                           | it looks like i have a 12 - day package available for ⟨unk⟩. ⟨unk⟩, would you be available to travel from the 27th of august to the 7th of september ? |
|                                           | ‘ is that for me and my friend |
| init reply                                 | would you like to upgrade to business class flights ? |
| final reply                                | sure you like me upgrade across business reviews flights ? |

Table 5: Example from NSE reply on text context.

| source                                      | and we have 3200 to spare between us |
| context                                     | i would like to depart from sapporo and arrive at punta cana . between august 29 and sept 1 this is a short trip for 13 adults |
|                                           | great ! just one adult traveller ? |
|                                           | and we have 3200 to spare between us |
| init reply                                 | when would you like to leave ? |
| final reply                                | excellent would you like to go ? |
Figure 13: Text context on switchboard dataset.
6 Conclusion & Discussion

The (qualitative) results show that the models are not really successful in dialogue. The larger part of tests resulted in ungrammatical replies. This is probably due to the small training sets. As language is very complex and the number of acceptable replies is huge, a reasonably large number of parameters is needed to capture generality, which is the case for the experiments. But the datasets are small and can be doubted to be big enough to have enough regularity to learn all of the parameters in the first way.

But since the hypotheses were not about language modeling ‘an sich’, but about information processing, I would argue something can be learned from the results, since the amount of possible information in one reply is far less than the number of possible replies, not as many examples are required as is the case for pure language modeling. Nevertheless, experiments on larger datasets are needed to be able to fully support the following claims.

In the following section the hypotheses are revisited, the last section some suggestions are done to further investigate the conclusions and questions.

6.1 Findings

Hypothesis 1 is mostly unconfirmed. When language and content modeling would be separable in an end-to-end system, a clear performance gain was to be expected from a hierarchical setting over standard seq2seq and thereby possibly of external memory over attention. Although the replies of the reasoning NSE do indicate a change in expression of a similar phenomenon (as exemplified in table 5), overall results do not indicate higher quality content when using a hierarchical setup or external memory.

Hypothesis 2 stated that one does not need the target DA to use DA information and is confirmed. The score of models using DA context is at least as high as that of models using the target DA. Note that the hypothesis is only minimally true, i.e. results from figure 11 show that it does not make any difference when modeling the context DA’s in a hierarchical content layer, opposed to providing the target to the decoder. But even in this form it implies that when DA information is desired in reply generation, it can be used ‘on the fly’, i.e. only the DA’s of the context need to be known which can be provided by a classifier module.

Furthermore, results of DA experiments show that DA information processing does not benefit from making use of external memory. So hypothesis 3 is confirmed. Neither did external memory prove to be better at context processing in general. So hypothesis 5 is unconfirmed.

Lastly, hypothesis 4 cannot be confirmed either, hierarchicality does not improve performance in all cases. In some cases it does though, and further research is required to investigate what the abilities of the three versions of content processors can do, as will be described promptly.
6.2 Future Work

The primary hypothesis \( h1 \) cannot be confirmed completely. So to explore the possibilities of disentangling language- and content-modeling with the NSE, a LM could be trained separately. This model can then be used to train a larger model, i.e. a pre-trained LM with the NSE fitted on top. Another possibility is to investigate how the capabilities of the DNC can be exploited in a similar manner. With an initial reply from a pre-trained LM evaluate each word separately (non-recursively) with a DNC which uses context as input. Hereby continuing on the findings of other work which showed hierarchically to be beneficial as well on the findings in this report indicated in the semantic changes the NSE made.
References


A Appendix

A.1 LSTM

The LSTM equations:

\[ i_t = \text{sigmoid}(W_{ii} x_t + b_{ii} + W_{hi} h_{(t-1)} + b_{hi}) \]  
\[ f_t = \text{sigmoid}(W_{if} x_t + b_{if} + W_{hf} h_{(t-1)} + b_{hf}) \]  
\[ g_t = \tanh(W_{ig} x_t + b_{ig} + W_{hc} h_{(t-1)} + b_{hg}) \]  
\[ o_t = \text{sigmoid}(W_{io} x_t + b_{io} + W_{ho} h_{(t-1)} + b_{ho}) \]  
\[ c_t = f_t \cdot c_{(t-1)} + i_t \cdot g_t \]  
\[ h_t = o_t \cdot \tanh(c_t) \]

![Figure 14: schematic](image)

A.2 Multi memory access NSE

The adapted equations for a decoder NSE using multiple encoded memories:

\[ o_t = f^{LSTM}_{r}(x_t) \]  
\[ z_t = \text{softmax}(o_t^T M_{t-1}) \]  
\[ m_{r,t} = z_t^T M_{t-1} \]  
\[ z^n_t = \text{softmax}(o_t^T M^n_{t-1}) \]  
\[ m^n_{r,t} = z^n_t^T M^n_{t-1} \]  
\[ c_t = f^{MLP}_{c}(o_t, m_{r,t}, m^n_{r,t}) \]  
\[ h_t = f^{LSTM}_{w}(c_t) \]  
\[ M_t = M_{t-1}(1_{l \times k} - z_t \otimes 1_{k \times 1})^T + (h_t \otimes 1_{l \times 1})(z_t \otimes 1_{k \times 1})^T \]  
\[ M^n_t = M^n_{t-1}(1_{l \times k} - z^n_t \otimes 1_{k \times 1})^T + (h_t \otimes 1_{l \times 1})(z^n_t \otimes 1_{k \times 1})^T \]