Recurrent Memory Networks for Language Modeling

Ke Tran Arianna Bisazza Christof Monz
Informatics Institute, University of Amsterdam
Science Park 904, 1098 XH Amsterdam, The Netherlands
{m.k.tran,a.bisazza,c.monz}@uva.nl

Abstract

Recurrent Neural Networks (RNNs) have obtained excellent result in many natural language processing (NLP) tasks. However, understanding and interpreting the source of this success remains a challenge. In this paper, we propose Recurrent Memory Network (RMN), a novel RNN architecture, that not only amplifies the power of RNN but also facilitates our understanding of its internal functioning and allows us to discover underlying patterns in data. We demonstrate the power of RMN on language modeling and sentence completion tasks. On language modeling, RMN outperforms Long Short-Term Memory (LSTM) network on three large German, Italian, and English dataset. Additionally we perform in-depth analysis of various linguistic dimensions that RMN captures. On Sentence Completion Challenge, for which it is essential to capture sentence coherence, our RMN obtains 69.2% accuracy, surpassing the previous state of the art by a large margin.[1]

1 Introduction

Recurrent Neural Networks (RNNs) (Elman, 1990; Mikolov et al., 2010) are remarkably powerful models for sequential data. Long Short-Term Memory (LSTM) (Hochreiter and Schmidhuber, 1997), a specific architecture of RNN, has a track record of success in many natural language processing tasks such as language modeling (Józefowicz et al., 2015), dependency parsing (Dyer et al., 2015), sentence compression (Filippova et al., 2015), and machine translation (Sutskever et al., 2014).

Within the context of natural language processing, a common assumption is that LSTMs are able to capture certain linguistic phenomena. Evidence supporting this assumption mainly comes from evaluating LSTMs in downstream applications: Bowman et al. (2015) carefully design two artificial datasets where sentences have explicit recursive structures. They show empirically that while processing the input linearly, LSTMs can implicitly exploit recursive structures of languages. Filippova et al. (2015) find that using explicit syntactic features within LSTMs in their sentence compression model hurts the performance of overall system. They then hypothesize that a basic LSTM is powerful enough to capture syntactic aspects which are useful for compression.

To understand and explain which linguistic dimensions are captured by an LSTM is non-trivial. This is due to the fact that the sequences of input histories are compressed into several dense vectors by the LSTM's components whose purposes with respect to representing linguistic information is not evident. To our knowledge, the only attempt to better understand the reasons of an LSTM's performance and limitations is the work of Karpathy et al. (2015) by means of visualization experiments and cell activation statistics in the context of character-level language modeling.

Our work is motivated by the difficulty in understanding and interpreting existing RNN architectures from a linguistic point of view. We propose Recurrent Memory Network (RMN), a novel RNN architecture that combines the strengths of both LSTM

[1] Our code and data are available at https://github.com/ketranm/RMN

Proceedings of NAACL-HLT 2016, pages 321–331,
San Diego, California, June 12-17, 2016.

and Memory Network (Sukhbaatar et al., 2015). In RMN, the Memory Block component—a variant of Memory Network—accesses the most recent input words and selectively attends to words that are relevant for predicting the next word given the current LSTM state. By looking at the attention distribution over history words, our RMN allows us not only to interpret the results but also to discover underlying dependencies present in the data.

In this paper, we make the following contributions:

1. We propose a novel RNN architecture that complements LSTM in language modeling. We demonstrate that our RMN outperforms competitive LSTM baselines in terms of perplexity on three large German, Italian, and English datasets.

2. We perform an analysis along various linguistic dimensions that our model captures. This is possible only because the Memory Block allows us to look into its internal states and its explicit use of additional inputs at each time step.

3. We show that, with a simple modification, our RMN can be successfully applied to NLP tasks other than language modeling. On the Sentence Completion Challenge (Zweig and Burges, 2012), our model achieves an impressive 69.2% accuracy, surpassing the previous state of the art 58.9% by a large margin.

2 Recurrent Neural Networks

Recurrent Neural Networks (RNNs) have shown impressive performances on many sequential modeling tasks due to their ability to encode unbounded input histories. However, training simple RNNs is difficult because of the vanishing and exploding gradient problems (Bengio et al., 1994; Pascanu et al., 2013). A simple and effective solution for exploding gradients is gradient clipping proposed by Pascanu et al. (2013). To address the more challenging problem of vanishing gradients, several variants of RNNs have been proposed. Among them, Long Short-Term Memory (Hochreiter and Schmidhuber, 1997) and Gated Recurrent Unit (Cho et al., 2014) are widely regarded as the most successful variants. In this work, we focus on LSTMs because they have been shown to outperform GRUs on language modeling tasks (Józefowicz et al., 2015). In the following, we will detail the LSTM architecture used in this work.

Long Short-Term Memory

Notation: Throughout this paper, we denote matrices, vectors, and scalars using bold uppercase (e. g., $\mathbf{W}$), bold lowercase (e. g., $\mathbf{b}$) and lowercase (e. g., α) letters, respectively.

The LSTM used in this work is specified as follows:

$$
\begin{aligned}
\mathbf{i}_t &= \text{sigm}(\mathbf{W}_{xi}\mathbf{x}_t + \mathbf{W}_{hi}\mathbf{h}_{t-1} + \mathbf{b}_i) \\
\mathbf{j}_t &= \text{sigm}(\mathbf{W}_{xj}\mathbf{x}_t + \mathbf{W}_{hj}\mathbf{h}_{t-1} + \mathbf{b}_j) \\
\mathbf{f}_t &= \text{sigm}(\mathbf{W}_{xf}\mathbf{x}_t + \mathbf{W}_{hf}\mathbf{h}_{t-1} + \mathbf{b}_f) \\
\mathbf{o}_t &= \tanh(\mathbf{W}_{xo}\mathbf{x}_t + \mathbf{W}_{ho}\mathbf{h}_{t-1} + \mathbf{b}_o) \\
\mathbf{c}_t &= \mathbf{c}_{t-1} \odot \mathbf{f}_t + \mathbf{i}_t \odot \mathbf{j}_t \\
\mathbf{h}_t &= \tanh(\mathbf{c}_t) \odot \mathbf{o}_t
\end{aligned}
$$

where $\mathbf{x}_t$ is the input vector at time step t, $\mathbf{h}_{t-1}$ is the LSTM hidden state at the previous time step, $\mathbf{W}_*$ and $\mathbf{b}_*$ are weights and biases. The symbol $\odot$ denotes the Hadamard product or element-wise multiplication.

Despite the popularity of LSTM in sequential modeling, its design is not straightforward to justify and understanding why it works remains a challenge (Hermans and Schrauwen, 2013; Chung et al., 2014; Greff et al., 2015; Józefowicz et al., 2015; Karpathy et al., 2015). There have been few recent attempts to understand the components of an LSTM from an empirical point of view: Greff et al. (2015) carry out a large-scale experiment of eight LSTM variants. The results from their 5,400 experimental runs suggest that forget gates and output gates are the most critical components of LSTMs. Józefowicz et al. (2015) conduct and evaluate over ten thousand RNN architectures and find that the initialization of the forget gate bias is crucial to the LSTM's performance. While these findings are important to help choosing appropriate LSTM architectures, they do not shed light on what information is captured by the hidden states of an LSTM.

Bowman et al. (2015) show that a vanilla LSTM, such as described above, performs reasonably well compared to a recursive neural network (Socher et al., 2011) that explicitly exploits tree structures on

two artificial datasets. They find that LSTMs can effectively exploit recursive structure in the artificial datasets. In contrast to these simple datasets containing a few logical operations in their experiments, natural languages exhibit highly complex patterns. The extent to which linguistic assumptions about syntactic structures and compositional semantics are reflected in LSTMs is rather poorly understood. Thus it is desirable to have a more principled mechanism allowing us to inspect recurrent architectures from a linguistic perspective. In the following section, we propose such a mechanism.

3 Recurrent Memory Network

It has been demonstrated that RNNs can retain input information over a long period. However, existing RNN architectures make it difficult to analyze what information is exactly retained at their hidden states at each time step, especially when the data has complex underlying structures, which is common in natural language. Motivated by this difficulty, we propose a novel RNN architecture called Recurrent Memory Network (RMN). On linguistic data, the RMN allows us not only to qualify which linguistic information is preserved over time and why this is the case but also to discover dependencies within the data (Section 5). Our RMN consists of two components: an LSTM and a *Memory Block* (MB) (Section 3.1). The MB takes the hidden state of the LSTM and compares it to the most recent inputs using an attention mechanism (Gregor et al., 2015; Bahdanau et al., 2014; Graves et al., 2014). Thus, analyzing the attention weights of a trained model can give us valuable insight into the information that is retained over time in the LSTM.

In the following, we describe in detail the MB architecture and the combination of the MB and the LSTM to form an RMN.

3.1 Memory Block

The Memory Block (Figure 1) is a variant of Memory Network (Sukhbaatar et al., 2015) with one hop (or a single-layer Memory Network). At time step t, the MB receives two inputs: the hidden state $\mathbf{h}_t$ of the LSTM and a set $\{x_i\}$ of n most recent words including the current word x_t. We refer to n as the memory size. Internally, the MB consists of

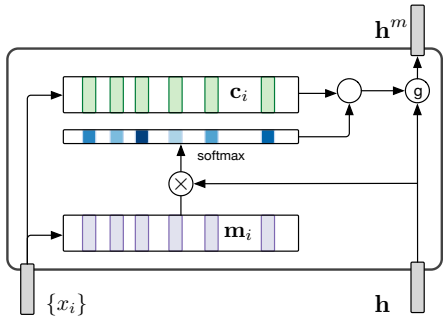

Figure 1: A graphical representation of the MB.

two lookup tables $\mathbf{M}$ and $\mathbf{C}$ of size $|V| \times d$, where $|V|$ is the size of the vocabulary. With a slight abuse of notation we denote $\mathbf{M}_i = \mathbf{M}(\{x_i\})$ and $\mathbf{C}_i = \mathbf{C}(\{x_i\})$ as $n \times d$ matrices where each row corresponds to an input memory embedding $\mathbf{m}_i$ and an output memory embedding $\mathbf{c}_i$ of each element of the set $\{x_i\}$. We use the matrix $\mathbf{M}_i$ to compute an attention distribution over the set $\{x_i\}$:

$$\mathbf{p}_t = \text{softmax}(\mathbf{M}_i \mathbf{h}_t) \tag{1}$$

When dealing with data that exhibits a strong temporal relationship, such as natural language, an additional temporal matrix $\mathbf{T} \in \mathbb{R}^{n \times d}$ can be used to bias attention with respect to the position of the data points. In this case, equation 1 becomes

$$\mathbf{p}_t = \text{softmax}\big((\mathbf{M}_i + \mathbf{T})\mathbf{h}_t\big) \tag{2}$$

We then use the attention distribution $\mathbf{p}_t$ to compute a context vector representation of $\{x_i\}$:

$$\mathbf{s}_t = \mathbf{C}_i^\mathsf{T} \mathbf{p}_t \tag{3}$$

Finally, we combine the context vector $\mathbf{s}_t$ and the hidden state $\mathbf{h}_t$ by a function $g(\cdot)$ to obtain the output $\mathbf{h}_t^m$ of the MB. Instead of using a simple addition function $g(\mathbf{s}_t, \mathbf{h}_t) = \mathbf{s}_t + \mathbf{h}_t$ as in Sukhbaatar et al. (2015), we propose to use a gating unit that decides how much it should trust the hidden state $\mathbf{h}_t$ and context $\mathbf{s}_t$ at time step t. Our gating unit is a form of Gated Recurrent Unit (Cho et al., 2014; Chung et al., 2014):

$$\mathbf{z}_t = \text{sigm}(\mathbf{W}_{sz}\mathbf{s}_t + \mathbf{U}_{hz}\mathbf{h}_t) \tag{4}$$

$$\mathbf{r}_t = \text{sigm}(\mathbf{W}_{sr}\mathbf{s}_t + \mathbf{U}_{hr}\mathbf{h}_t) \tag{5}$$

$$\tilde{\mathbf{h}}_t = \tanh(\mathbf{W}\mathbf{s}_t + \mathbf{U}(\mathbf{r}_t \odot \mathbf{h}_t)) \tag{6}$$

$$\mathbf{h}_t^m = (\mathbf{1} - \mathbf{z}_t) \odot \mathbf{h}_t + \mathbf{z}_t \odot \tilde{\mathbf{h}}_t \tag{7}$$

where $\mathbf{z}_t$ is an update gate, $\mathbf{r}_t$ is a reset gate.

The choice of the composition function $g(\cdot)$ is crucial for the MB especially when one of its input comes from the LSTM. The simple addition function might overwrite the information within the LSTM's hidden state and therefore prevent the MB from keeping track of information in the distant past. The gating function, on the other hand, can control the degree of information that flows from the LSTM to the MB's output.

3.2 RMN Architectures

As explained above, our proposed MB receives the hidden state of the LSTM as one of its input. This leads to an intuitive combination of the two units by stacking the MB on top of the LSTM. We call this architecture Recurrent-Memory (RM). The RM architecture, however, does not allow interaction between Memory Blocks at different time steps. To enable this interaction we can stack one more LSTM layer on top of the RM. We call this architecture Recurrent-Memory-Recurrent (RMR).

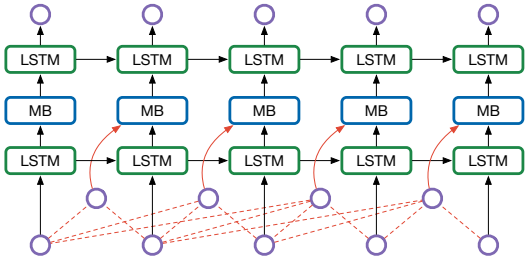

Figure 2: A graphical illustration of an unfolded RMR with memory size 4. Dashed line indicates concatenation. The MB takes the output of the bottom LSTM layer and the 4-word history as its input. The output of the MB is then passed to the second LSTM layer on top. There is no direct connection between MBs of different time steps. The last LSTM layer carries the MB's outputs recurrently.

4 Language Model Experiments

Language models play a crucial role in many NLP applications such as machine translation and speech recognition. Language modeling also serves as a standard test bed for newly proposed models (Sukhbaatar et al., 2015; Kalchbrenner et al., 2015). We conjecture that, by explicitly accessing history words, RMNs will offer better predictive power than the existing recurrent architectures. We therefore evaluate our RMN architectures against state-of-the-art LSTMs in terms of perplexity.

4.1 Data

We evaluate our models on three languages: English, German, and Italian. We are especially interested in German and Italian because of their larger vocabularies and complex agreement patterns. Table 1 summarizes the data used in our experiments.

Lang	Train	Dev	Test	$\lvert s\rvert$	$\lvert V\rvert$
En	26M	223K	228K	26	77K
De	22M	202K	203K	22	111K
It	29M	207K	214K	29	104K

Table 1: Data statistics. $\lvert s\rvert$ denotes the average sentence length and $\lvert V\rvert$ the vocabulary size.

The training data correspond to approximately 1M sentences in each language. For English, we use all the News Commentary data (8M tokens) and 18M tokens from News Crawl 2014 for training. Development and test data are randomly drawn from the concatenation of the WMT 2009-2014 test sets (Bojar et al., 2015). For German, we use the first 6M tokens from the News Commentary data and 16M tokens from News Crawl 2014 for training. For development and test data we use the remaining part of the News Commentary data concatenated with the WMT 2009-2014 test sets. Finally, for Italian, we use a selection of 29M tokens from the PAISÀ corpus (Lyding et al., 2014), mainly including Wikipedia pages and, to a minor extent, Wikibooks and Wikinews documents. For development and test we randomly draw documents from the same corpus.

4.2 Setup

Our baselines are a 5-gram language model with Kneser-Ney smoothing, a Memory Network (MemN) (Sukhbaatar et al., 2015), a vanilla single-layer LSTM, and two stacked LSTMs with two and three layers respectively. N-gram models have been used intensively in many applications for their excellent performance and fast training. Chen et al. (2015) show that n-gram model outperforms a popular feed-forward language model (Bengio et al.,

2003) on a one billion word benchmark (Chelba et al., 2013). While taking longer time to train, RNNs have been proven superior to n-gram models.

We compare these baselines with our two model architectures: RMR and RM. For each of our models, we consider two settings: with or without temporal matrix (+tM or –tM), and linear vs. gating composition function. In total, we experiment with eight RMN variants.

For all neural network models, we set the dimension of word embeddings, the LSTM hidden states, its gates, the memory input, and output embeddings to 128. The memory size is set to 15. The bias of the LSTM's forget gate is initialized to 1 (Józefowicz et al., 2015) while all other parameters are initialized uniformly in $(-0.05, 0.05)$. The initial learning rate is set to 1 and is halved at each epoch after the forth epoch. All models are trained for 15 epochs with standard stochastic gradient descent (SGD). During training, we rescale the gradients whenever their norm is greater than 5 (Pascanu et al., 2013).

Sentences with the same length are grouped into buckets. Then, mini-batches of 20 sentences are drawn from each bucket. We do not use truncated back-propagation through time, instead gradients are fully back-propagated from the end of each sentence to its beginning. When feeding in a new mini-batch, the hidden states of LSTMs are reset to zeros, which ensures that the data is properly modeled at the sentence level. For our RMN models, instead of using padding, at time step $t < n$, we use a slice $\mathbf{T}[1:t] \in \mathbb{R}^{t \times d}$ of the temporal matrix $\mathbf{T} \in \mathbb{R}^{n \times d}$.

4.3 Results

Perplexities on the test data are given in Table 2. All RMN variants largely outperform n-gram and MemN models, and most RMN variants also outperform the competitive LSTM baselines. The best results overall are obtained by RM with temporal matrix and gating composition (+tM-g).

Our results agree with the hypothesis of mitigating prediction error by explicitly using the last n words in RNNs (Karpathy et al., 2015). We further observe that using a temporal matrix always benefits the RM architectures. This can be explained by seeing the RM as a principled way to combine an LSTM and a neural n-gram model. By contrast, RMR works better without temporal matrix but its overall performance is not as good as RM. This suggests that we need a better mechanism to address the interaction between MBs, which we leave to future work. Finally, the proposed gating composition function outperforms the linear one in most cases.

Model		**De**	**It**	**En**
5-gram	–	225.8	167.5	219.0
MemN	1 layer	169.3	127.5	188.2
LSTM	1 layer	135.8	108.0	145.1
	2 layers	128.6	105.9	139.7
	3 layers	125.1	106.5	136.6
RMR	+tM-l	127.5	109.9	133.3
	–tM-l	126.4	106.1	134.5
	+tM-g	126.2	99.5	135.2
	–tM-g	**122.0**	**98.6**	**131.2**
RM	+tM-l	121.5	92.4	**127.2**
	–tM-l	122.9	94.0	130.4
	+tM-g	**118.6**	**88.9**	128.8
	–tM-g	129.7	96.6	135.7

Table 2: Perplexity comparison including RMN variants with and without temporal matrix (tM) and linear (l) versus gating (g) composition function.

For historical reasons, we also run a stacked three-layer LSTM and a RM(+tM-g) on the much smaller Penn Treebank dataset (Marcus et al., 1993) with the same setting described above. The respective perplexities are 126.1 and 123.5.

5 Attention Analysis

The goal of our RMN design is twofold: (i) to obtain better predictive power and (ii) to facilitate understanding of the model and discover patterns in data. In Section 4, we have validated the predictive power of the RMN and below we investigate the source of this performance based on linguistic assumptions of word co-occurrences and dependency structures.

5.1 Positional and lexical analysis

As a first step towards understanding RMN, we look at the average attention weights of each history word position in the MB of our two best model variants (Figure 3). One can see that the attention mass tends to concentrate at the rightmost position (the current

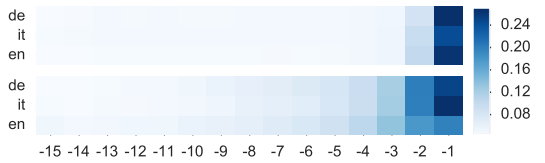

Figure 3: Average attention per position of RMN history. Top: RMR(–tM-g), bottom: RM(+tM-g). Rightmost positions represent most recent history.

word) and decreases when moving further to the left (less recent words). This is not surprising since the success of n-gram language models has demonstrated that the most recent words provide important information for predicting the next word. Between the two variants, the RM average attention mass is less concentrated to the right. This can be explained by the absence of an LSTM layer on top, meaning that the MB in the RM architecture has to pay more attention to the more distant words in the past. The remaining analyses described below are performed on the RM(+tM-g) architecture as this yields the best perplexity results overall.

Beyond average attention weights, we are interested in those cases where attention focuses on distant positions. To this end, we randomly sample 100 words from test data and visualize attention distributions over the last 15 words. Figure 4 shows the attention distributions for random samples of German and Italian. Again, in many cases attention weights concentrate around the last word (bottom row). However, we observe that many long distance words also receive noticeable attention mass. Interestingly, for many predicted words, attention is distributed evenly over memory positions, possibly indicating cases where the LSTM state already contains enough information to predict the next word.

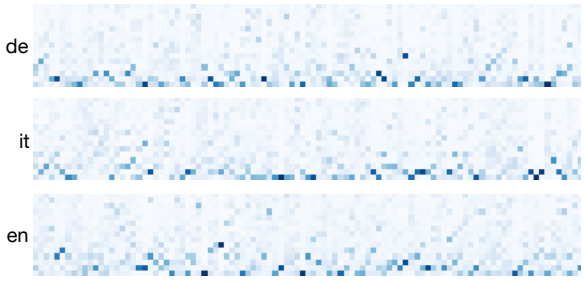

Figure 4: Attention visualization of 100 word samples. Bottom positions in each plot represent most recent history. Darker color means higher weight.

To explain the long-distance dependencies, we first hypothesize that our RMN mostly memorizes frequent co-occurrences. We run the RM(+tM-g) model on the German development and test sentences, and select those pairs of (*most-attended-word, word-to-predict*) where the MB's attention concentrates on a word more than six positions to the left. Then, for each set of pairs with equal distance, we compute the mean frequency of corresponding co-occurrences seen in the training data (Table 3). The lack of correlation between frequency and memory location suggests that RMN does more than simply memorizing frequent co-occurrences.

d	7	8	9	10	11	12	13	14	15
μ	54	63	42	67	87	47	67	44	24

Table 3: Mean frequency (μ) of (*most-attended-word, word-to-predict*) pairs grouped by relative distance (d).

Previous work (Hermans and Schrauwen, 2013; Karpathy et al., 2015) studied this property of LSTMs by analyzing simple cases of closing brackets. By contrast RMN allows us to discover more interesting dependencies in the data. We manually inspect those high-frequency pairs to see whether they display certain linguistic phenomena. We observe that RMN captures, for example, *separable verbs* and *fixed expressions* in German. Separable verbs are frequent in German: they typically consist of preposition+verb constructions, such *ab+hängen* ('to depend') or *aus+schließen* ('to exclude'), and can be spelled together (*abhängen*) or apart as in '*hängen von der Situation ab*' ('depend on the situation'), depending on the grammatical construction. Figure 5a shows a long-dependency example for the separable verb *abhängen (to depend)*. When predicting the verb's particle *ab*, the model correctly attends to the verb's core *hängt* occurring seven words to the left. Figure 5b and 5c show fixed expression examples from German and Italian, respectively: *schlüsselrolle ... spielen (play a key role)* and *insignito ... titolo (awarded title)*. Here too, the model correctly attends to the key word despite its long distance from the word to predict.

Figure 5: Examples of distant memory positions attended by RMN. The resulting top five word predictions are shown with the respective log-probabilities. The correct choice (in bold) was ranked first in sentences (a,b) and second in (c).

Other interesting examples found by the RMN in the test data include:

German: findet *statt* (takes *place*), kehrte *zurück* (came *back*), fragen *antworten* (questions *answers*), kämpfen *gegen* (fight *against*), bleibt *erhalten* (remains *intact*), verantwortung *übernimmt* (*takes* responsibility);

Italian: sinistra *destra* (left *right*), latitudine *longitudine* (latitude *longitude*), collegata *tramite* (connected *through*), sposò *figli* (got-married *children*), insignito *titolo* (awarded *title*).

5.2 Syntactic analysis

It has been conjectured that RNNs, and LSTMs in particular, model text so well because they capture syntactic structure implicitly. Unfortunately this has been hard to prove, but with our RMN model we can get closer to answering this important question.

We produce dependency parses for our test sets using (Sennrich et al., 2013) for German and (Attardi et al., 2009) for Italian. Next we look at how much attention mass is concentrated by the RM(+tM-g) model on different dependency types. Figure 6 shows, for each language, a selection of ten dependency types that are often long-distance.[2] Dependency direction is marked by an arrow: e.g. $\rightarrow$*mod* means that the word to predict is a modifier of the attended word, while *mod*$\leftarrow$ means that the attended word is a modifier of the word to predict.[3] White cells denote combinations of position and dependency type that were not present in the test data.

While in most of the cases closest positions are attended the most, we can see that some dependency types also receive noticeably more attention than the average (ALL) on the long-distance positions. In German, this is mostly visible for the head of separable verb particles ($\rightarrow$*avz*), which nicely supports our observations in the lexical analysis (Section 5.1). Other attended dependencies include: auxiliary verbs ($\rightarrow$*aux*) when predicting the second element of a complex tense (*hat . . . gesagt / has said*); subordinating conjunctions (*konj*$\leftarrow$) when predicting the clause-final inflected verb (*dass sie sagen sollten / that they should say*); control verbs ($\rightarrow$*obji*) when predicting the infinitive verb (*versucht ihr zu helfen / tries to help her*). Out of the Italian dependency types selected for their frequent long-distance occurrences (bottom of Figure 6), the most attended are argument heads ($\rightarrow$*arg*), complement heads ($\rightarrow$*comp*), object heads ($\rightarrow$*obj*) and subjects (*subj*$\leftarrow$). This suggests that RMN is mainly capturing predicate argument structure in Italian. Notice that syntactic annotation is never used to train the model, but only to analyze its predictions.

We can also use RMN to discover which complex dependency paths are important for word prediction. To mention just a few examples, high attention on

[2]The full plots are available at `https://github.com/ketranm/RMN`. The German and Italian tag sets are explained in (Simi et al., 2014) and (Foth, 2006) respectively.

[3]Some dependency directions, like *obj*$\leftarrow$ in Italian, are almost never observed due to order constraints of the language.

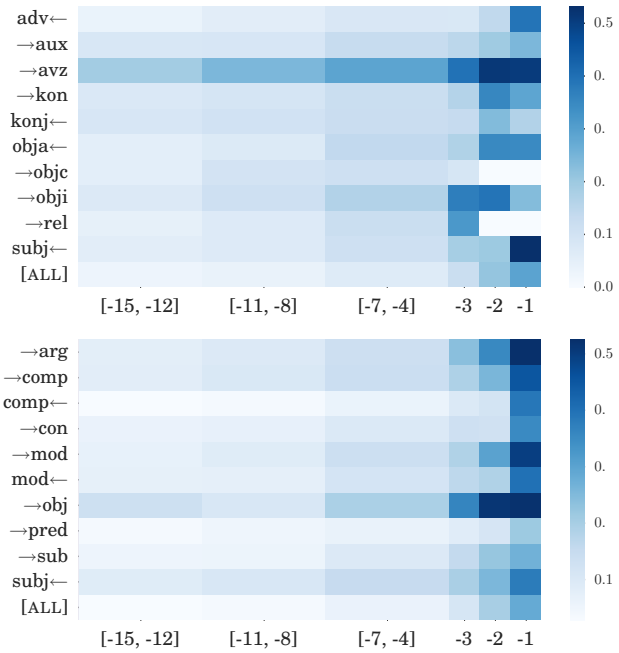

Figure 6: Average attention weights per position, broken down by dependency relation type+direction between the attended word and the word to predict. Top: German. Bottom: Italian. More distant positions are binned.

the German path *[subj←,→kon,→cj]* indicates that the model captures morphological agreement between coordinate clauses in non-trivial constructions of the kind: *spielen die Kinder im Garten und singen / the children play in the garden and sing*. In Italian, high attention on the path *[→obj,→comp,→prep]* denotes cases where the semantic relatedness between a verb and its object does not stop at the object's head, but percolates down to a prepositional phrase attached to it (*passò buona parte della sua vita / spent a large part of his life*). Interestingly, both local n-gram context and immediate dependency context would have missed these relations.

While much remains to be explored, our analysis shows that RMN discovers patterns far more complex than pairs of opening and closing brackets, and suggests that the network's hidden state captures to a large extent the underlying structure of text.

6 Sentence Completion Challenge

The Microsoft Research Sentence Completion Challenge (Zweig and Burges, 2012) has recently become a test bed for advancing statistical language modeling. We choose this task to demonstrate the effectiveness of our RMN in capturing sentence coherence. The test set consists of 1,040 sentences selected from five Sherlock Holmes novels by Conan Doyle. For each sentence, a content word is removed and the task is to identify the correct missing word among five given candidates. The task is carefully designed to be non-solvable for local language models such as n-gram models. The best reported result is 58.9% accuracy (Mikolov et al., 2013)[4] which is far below human accuracy of 91% (Zweig and Burges, 2012).

As baseline we use a stacked three-layer LSTM. Our models are two variants of RM(+tM-g), each consisting of three LSTM layers followed by a MB. The first variant (unidirectional-RM) uses n words preceding the word to predict, the second (bidirectional-RM) uses the n words preceding *and* the n words following the word to predict, as MB input. We include bidirectional-RM in the experiments to show the flexibility of utilizing future context in RMN.

We train all models on the standard training data of the challenge, which consists of 522 novels from Project Gutenberg, preprocessed similarly to (Mnih and Kavukcuoglu, 2013). After sentence splitting, tokenization and lowercasing, we randomly select 19,000 sentences for validation. Training and validation sets include 47M and 190K tokens respectively. The vocabulary size is about 64,000.

We initialize and train all the networks as described in Section 4.2. Moreover, for regularization, we place dropout (Srivastava et al., 2014) after each LSTM layer as suggested in (Pham et al., 2014). The dropout rate is set to 0.3 in all the experiments.

Table 4 summarizes the results. It is worth to mention that our LSTM baseline outperforms a dependency RNN making explicit use of syntactic information (Mirowski and Vlachos, 2015) and performs on par with the best published result (Mikolov et al., 2013). Our unidirectional-RM sets a new state of the art for the Sentence Completion Challenge with 69.2% accuracy. Under the same setting of d we observe that using bidirectional context does not

[4]The authors use a weighted combination of skip-ngram and RNN without giving any technical details.

The stage lost a fine ____ , even as science lost an acute reasoner , when he became a specialist in crime
a) linguist b) hunter c) **actor**$^{\clubsuit}$ d) estate e) horseman$^{\diamondsuit}$

What passion of hatred can it be which leads a man to ____ in such a place at such a time
a) **lurk**$^{\clubsuit}$ b) dine$^{\diamondsuit}$ c) luxuriate d) grow e) wiggle

My heart is ____ already since i have confided my trouble to you
a) falling b) distressed$^{\diamondsuit}$ c) soaring d) **lightened**$^{\clubsuit}$ e) punished

My morning's work has not been ____ , since it has proved that he has the very strongest motives for standing in the way of anything of the sort
a) invisible b) neglected$^{\diamondsuit\clubsuit}$ c) overlooked d) **wasted** e) deliberate

That is his ____ fault , but on the whole he's a good worker
a) **main** b) successful c) mother's$^{\clubsuit}$ d) generous e) favourite$^{\diamondsuit}$

Figure 7: Examples of sentence completion. The correct option is in boldface. Predictions by the LSTM baseline and by our best RMN model are marked by $^{\diamondsuit}$ and $^{\clubsuit}$ respectively.

Model	n	d	**Accuracy**
LSTM	–	256	56.0
unidirectional-RM	15	256	64.3
	15	512	**69.2**
bidirectional-RM	7	256	59.6
	10	512	67.0

Table 4: Accuracy on 1,040 test sentences. We use perplexity to choose the best model. Dimension of word embeddings, LSTM hidden states, and gate g parameters are set to d.

bring additional advantage to the model. Mnih and Kavukcuoglu (2013) also report a similar observation. We believe that RMN may achieve further improvements with hyper-parameter optimization.

Figure 7 shows some examples where our best RMN beats the already very competitive LSTM baseline, or where both models fail. We can see that in some sentences the necessary clues to predict the correct word occur only to its *right*. While this seems to conflict with the worse result obtained by the bidirectional-RM, it is important to realize that prediction corresponds to the whole sentence probability. Therefore a badly chosen word can have a negative effect on the score of future words. This appears to be particularly true for the RMN due to its ability to directly access (distant) words in the history. The better performance of unidirectional versus bidirectional-RM may indicate that the attention in the memory block can be distributed reliably only on words that have been already seen and summarized by the current LSTM state. In future work, we may investigate whether different ways to combine two RMNs running in opposite directions further improve accuracy on this challenging task.

7 Conclusion

We have proposed the Recurrent Memory Network (RMN), a novel recurrent architecture for language modeling. Our RMN outperforms LSTMs in terms of perplexity on three large dataset and allows us to analyze its behavior from a linguistic perspective. We find that RMNs learn important co-occurrences regardless of their distance. Even more interestingly, our RMN implicitly captures certain dependency types that are important for word prediction, despite being trained without any syntactic information. Finally RMNs obtain excellent performance at modeling sentence coherence, setting a new state of the art on the challenging sentence completion task.

Acknowledgments

This research was funded in part by the Netherlands Organization for Scientific Research (NWO) under project numbers 639.022.213 and 612.001.218.

References

Giuseppe Attardi, Felice Dell'Orletta, Maria Simi, and Joseph Turian. 2009. Accurate dependency parsing with a stacked multilayer perceptron. In *Proceedings of Evalita'09, Evaluation of NLP and Speech Tools for Italian*, Reggio Emilia, Italy.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. In *ICLR 2015*, San Diego, CA, USA, May.

Yoshua Bengio, Patrice Simard, and Paolo Frasconi. 1994. Learning long-term dependencies with gradient descent is difficult. *Transaction on Neural Networks*, 5(2):157–166, March.

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin. 2003. A neural probabilistic language model. *J. Mach. Learn. Res.*, 3:1137–1155, March.

Ondřej Bojar, Rajen Chatterjee, Christian Federmann, Barry Haddow, Matthias Huck, Chris Hokamp, Philipp Koehn, Varvara Logacheva, Christof Monz, Matteo Negri, Matt Post, Carolina Scarton, Lucia Specia, and Marco Turchi. 2015. Findings of the 2015 workshop on statistical machine translation. In *Proceedings of the Tenth Workshop on Statistical Machine Translation*, pages 1–46, Lisbon, Portugal, September. Association for Computational Linguistics.

Samuel R. Bowman, Christopher D. Manning, and Christopher Potts. 2015. Tree-structured composition in neural networks without tree-structured architectures. In *Proceedings of Proceedings of the NIPS 2015 Workshop on Cognitive Computation: Integrating Neural and Symbolic Approaches*, December.

Ciprian Chelba, Tomas Mikolov, Mike Schuster, Qi Ge, Thorsten Brants, Phillipp Koehn, and Tony Robinson. 2013. One billion word benchmark for measuring progress in statistical language modeling. Technical report, Google.

Welin Chen, David Grangier, and Michael Auli. 2015. Strategies for Training Large Vocabulary Neural Language Models. *ArXiv e-prints*, December.

Kyunghyun Cho, Bart van Merrienboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014. On the properties of neural machine translation: Encoder–decoder approaches. In *Proceedings of SSST-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation*, pages 103–111, Doha, Qatar, October. Association for Computational Linguistics.

Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. In *NIPS Deep Learning and Representation Learning Workshop*.

Chris Dyer, Miguel Ballesteros, Wang Ling, Austin Matthews, and Noah A. Smith. 2015. Transition-based dependency parsing with stack long short-term memory. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*, pages 334–343, Beijing, China, July. Association for Computational Linguistics.

Jeffrey L. Elman. 1990. Finding structure in time. *Cognitive Science*, 14(2):179–211.

Katja Filippova, Enrique Alfonseca, Carlos A. Colmenares, Lukasz Kaiser, and Oriol Vinyals. 2015. Sentence compression by deletion with lstms. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 360–368, Lisbon, Portugal, September. Association for Computational Linguistics.

Kilian A. Foth. 2006. *Eine umfassende Constraint-Dependenz-Grammatik des Deutschen*. Fachbereich Informatik.

Alex Graves, Greg Wayne, and Ivo Danihelka. 2014. Neural turing machines. *CoRR*, abs/1410.5401.

Klaus Greff, Rupesh Kumar Srivastava, Jan Koutník, Bas R. Steunebrink, and Jürgen Schmidhuber. 2015. LSTM: A search space odyssey. *CoRR*, abs/1503.04069.

Karol Gregor, Ivo Danihelka, Alex Graves, Danilo Jimenez Rezende, and Daan Wierstra. 2015. DRAW: A recurrent neural network for image generation. In *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, pages 1462–1471.

Michiel Hermans and Benjamin Schrauwen. 2013. Training and analysing deep recurrent neural networks. In C.J.C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 190–198. Curran Associates, Inc.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation*, 9(8):1735–1780, November.

Rafal Józefowicz, Wojciech Zaremba, and Ilya Sutskever. 2015. An empirical exploration of recurrent network architectures. In *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, pages 2342–2350.

Nal Kalchbrenner, Ivo Danihelka, and Alex Graves. 2015. Grid long short-term memory. *CoRR*, abs/1507.01526.

Andrej Karpathy, Justin Johnson, and Fei-Fei Li. 2015. Visualizing and understanding recurrent networks. *CoRR*, abs/1506.02078.

Verena Lyding, Egon Stemle, Claudia Borghetti, Marco Brunello, Sara Castagnoli, Felice Dell'Orletta, Henrik

Dittmann, Alessandro Lenci, and Vito Pirrelli. 2014. The PAISÀ corpus of italian web texts. In *Proceedings of the 9th Web as Corpus Workshop (WaC-9)*, pages 36–43, Gothenburg, Sweden, April. Association for Computational Linguistics.

Mitchell P. Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. 1993. Building a large annotated corpus of english: The penn treebank. *Comput. Linguist.*, 19(2):313–330, June.

Tomas Mikolov, Martin Karafiát, Lukás Burget, Jan Cernocký, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In *INTERSPEECH 2010, 11th Annual Conference of the International Speech Communication Association, Makuhari, Chiba, Japan, September 26-30, 2010*, pages 1045–1048.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. In *ICLR*.

Piotr Mirowski and Andreas Vlachos. 2015. Dependency recurrent neural language models for sentence completion. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 511–517, Beijing, China, July. Association for Computational Linguistics.

Andriy Mnih and Koray Kavukcuoglu. 2013. Learning word embeddings efficiently with noise-contrastive estimation. In C.J.C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 2265–2273. Curran Associates, Inc.

Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. 2013. On the difficulty of training recurrent neural networks. In *ICML (3)*, volume 28 of *JMLR Proceedings*, pages 1310–1318.

Vu Pham, Christopher Bluche, Théodore Kermorvant, and Jérôme Louradour. 2014. Dropout improves recurrent neural networks for handwriting recognition. In *International Conference on Frontiers in Handwriting Recognition (ICFHR)*, pages 285–290, Sept.

Rico Sennrich, Martin Volk, and Gerold Schneider. 2013. Exploiting synergies between open resources for german dependency parsing, pos-tagging, and morphological analysis. In *Recent Advances in Natural Language Processing (RANLP 2013)*, pages 601–609, September.

Maria Simi, Cristina Bosco, and Simonetta Montemagni. 2014. Less is more? towards a reduced inventory of categories for training a parser for the italian stanford dependencies. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*, Reykjavik, Iceland, may. European Language Resources Association (ELRA).

Richard Socher, Jeffrey Pennington, Eric H. Huang, Andrew Y. Ng, and Christopher D. Manning. 2011. Semi-supervised recursive autoencoders for predicting sentiment distributions. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '11, pages 151–161, Stroudsburg, PA, USA. Association for Computational Linguistics.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.*, 15(1):1929–1958, January.

Sainbayar Sukhbaatar, Arthur Szlam, Jason Weston, and Rob Fergus. 2015. End-to-end memory networks. In C. Cortes, N.D. Lawrence, D.D. Lee, M. Sugiyama, R. Garnett, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 2431–2439. Curran Associates, Inc.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In Z. Ghahramani, M. Welling, C. Cortes, N.D. Lawrence, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 3104–3112. Curran Associates, Inc.

Geoffrey Zweig and Chris J. C. Burges. 2012. A challenge set for advancing language modeling. In *Proceedings of the NAACL-HLT 2012 Workshop: Will We Ever Really Replace the N-gram Model? On the Future of Language Modeling for HLT*, WLM '12, pages 29–36, Stroudsburg, PA, USA. Association for Computational Linguistics.