Cut to the Chase: A Context Zoom-in Network for Reading Comprehension

Sathish Indurthi[1] **Seunghak Yu**[1,*] **Seohyun Back**[1] **Heriberto Cuayáhuitl**[1,2]

[1] Samsung Research, Seoul, Korea
[2] School of Computer Science, University of Lincoln, Lincoln, United Kingdom
{s.indurthi, seunghak.yu, scv.back}@samsung.com
HCuayahuitl@lincoln.ac.uk

Abstract

In recent years many deep neural networks have been proposed to solve Reading Comprehension (RC) tasks. Most of these models suffer from reasoning over long documents and do not trivially generalize to cases where the answer is not present as a span in a given document. We present a novel neural-based architecture that is capable of extracting relevant regions based on a given question-document pair and generating a well-formed answer. To show the effectiveness of our architecture, we conducted several experiments on the recently proposed and challenging RC dataset 'NarrativeQA'. The proposed architecture outperforms state-of-the-art results (Tay et al., 2018) by 12.62% (ROUGE-L) relative improvement.

1 Introduction

Building Artificial Intelligence (AI) algorithms to teach machines to read and to comprehend text is a long-standing challenge in Natural Language Processing (NLP). A common strategy for assessing these AI algorithms is by treating them as RC tasks. This can be formulated as finding an answer to a question given the document(s) as evidence. Recently, many deep-learning based models (Seo et al., 2017; Xiong et al., 2017; Wang et al., 2017; Shen et al., 2017; Clark and Gardner, 2017) have been proposed to solve RC tasks based on the SQuAD (Rajpurkar et al., 2016) and TriviaQA (Joshi et al., 2017) datasets, reaching human level performance. A common approach in these models is to score and/or extract candidate spans conditioned on a given question-document pair.

Most of these models have limited applicability to real problems for the following reasons. They do not generalize well to scenarios where the answer is not present as a span, or where several discontinuous parts of the document are required to form the answer. In addition, unlike humans, they can not easily skip through irrelevant parts to comprehend long documents (Masson, 1983).

To address the issues above we develop a novel context zoom-in network (ConZNet) for RC tasks, which can skip through irrelevant parts of a document and generate an answer using only the relevant regions of text. The ConZNet architecture consists of two phases. In the first phase we identify the relevant regions of text by employing a reinforcement learning algorithm. These relevant regions are not only useful to generate the answer, but can also be presented to the user as supporting information along with the answer. The second phase is based on an encoder-decoder architecture, which comprehends the identified regions of text and generates the answer by using a residual self-attention network as encoder and a RNN-based sequence generator along with a pointer network (Vinyals et al., 2015) as the decoder. It has the ability to generate better well-formed answers not verbatim present in the document than span prediction models.

Recently, there have been several attempts to adopt condensing documents in RC tasks. Wang et al. (2018) retrieve a relevant paragraph based on the question and predict the answer span. Choi et al. (2017) select sentence(s) to make a summary of the entire document with a feed-forward network and generate an answer based on the summary. Unlike existing approaches, our method has the ability to select relevant regions of text not just based on the question but also on how well regions are related to each other. Moreover, our decoder combines span prediction and sequence generation. This allows the decoder to copy words from the relevant regions of text as well as to generate words from a fixed vocabulary.

We evaluate our model using one of the challenging RC datasets, called 'NarrativeQA', which

[*] To whom correspondence should be addressed.

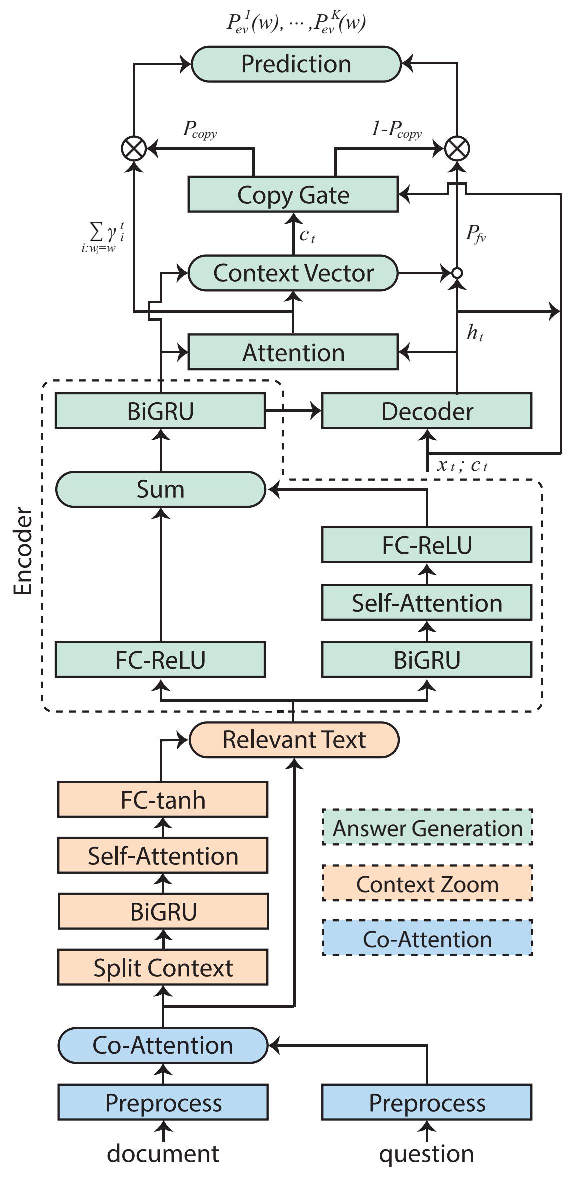

Figure 1: The proposed ConZNet architecture

was released recently by Kočiskỳ et al. (2017). Experimental results show the usefulness of our framework for RC tasks and we outperform state-of-the-art results on this dataset.

2 Proposed Architecture

An overview of our architecture is shown in Figure 1, which consists of two phases. First, the identification of relevant regions of text is computed by the **Co-attention** and **Context Zoom** layers as explained in Sections 2.1 and 2.2. Second, the comprehension of identified regions of text and output generation is computed by **Answer Generation** block as explained in Section 2.3.

2.1 Co-attention layer

The words in the document, question and answer are represented using pre-trained word embeddings (Pennington et al., 2014). These word-based embeddings are concatenated with their corresponding char embeddings. The char embeddings are learned by feeding all the characters of a word into a Convolutional Neural Network (CNN) (Kim, 2014). We further encode the document and question embeddings using a shared bi-directional GRU (Cho et al., 2014) to get context-aware representations.

We compute the co-attention between document and question to get question-aware representations for the document by using tri-linear attention as proposed by Seo et al. (2017). Let d_i be the vector representation for the document word i, q_j be the vector for the question word j, and l_d and l_q be the lengths of the document and question respectively. The tri-linear attention is calculated as

$$a_{ij} = w_d d_i + w_q q_j + w_{dq}(d_i \odot q_j), \qquad (1)$$

where w_d, w_q, and w_{dq} are learnable parameters and $\odot$ denotes the element-wise multiplication.

We compute the attended document word $\tilde{d}_i$ by first computing $\lambda_i = softmax(a_{i:})$ and followed by $\tilde{d}_i = \sum_{j=1}^{l_q} \lambda_{ij} q_j$. Similarly, we compute a question to document attention vector $\tilde{q}$ by first computing $b = softmax(\max(a_{i:}))$ and followed by $\tilde{q} = \sum_{i=i}^{l_d} d_i b_i$. Finally, d_i, $\tilde{d}_i$, $d_i \odot \tilde{d}_i$, $\tilde{d}_i \odot \tilde{q}$ are concatenated to yield a query-aware contextual representation for each word in the document.

2.2 Context Zoom Layer

This layer finds relevant regions of text. We use reinforcement learning to do that, with the goal of improving answer generation accuracy – see Section 2.4.

The *Split Context* operation splits the attended document vectors into sentences or fixed size chunks (useful when sentence tokenization is not available for a particular language). This results in n text regions with each having length l_k, where $l_d = \sum_{k=1}^{n} l_k$. We then get the representations, denoted as z_k, for each text region by running a *BiGRU* and concatenating the last states of the forward and backward GRUs.

The text region representations, z_k, encode how well they are related to the question, and their surrounding context. Generating an answer may depend on multiple regions, and it is important for each text region to collect cues from other regions which are outside of their surroundings. We can compute this by using a *Self-Attention* layer. It is a special case of co-attention where both operands (d_i and q_j) are the text fragment itself, computed by setting $a_{ij} = -\infty$ when $i = j$ in Eq. 1.

These further self-attended text region representations, $\tilde{z}_k$, are passed through a linear layer with *tanh* activation and *softmax* layer as follows:

$$u = tanh(W_c[\tilde{z}_1, \cdots, \tilde{z}_n] + b_c), \quad (2)$$

$$\psi = softmax(u), \quad (3)$$

where ψ is the probability distribution of text regions, which is the evidence used to generate the answer. The policy of the reinforcement learner is defined as $\pi(r|u;\theta_z) = \psi_r$, where ψ_r is the probability of a text region r (agent's action) being selected, u is the environment state as defined in Eq. 2, and θ_z are the learnable parameters. During the training time we sample text regions using ψ, in inference time we follow greedy evaluation by selecting most probable region(s).

2.3 Answer Generation

This component is implemented based on the encoder-decoder architecture of (Sutskever et al., 2014). The selected text regions from the *Context Zoom* layer are given as input to the encoder, where its output is given to the decoder in order to generate the answer.

The encoder block uses residual connected self-attention layer followed by a BiGRU. The selected relevant text regions ($\in \psi_r$) are first passed through a separate BiGRU, then we apply a self-attention mechanism similar to the *Context Zoom* layer followed by a linear layer with ReLU activations. The encoder's output consists of representations of the relevant text regions, denoted by e_i.

The decoder block is based on an attention mechanism (Bahdanau et al., 2015) and a copy mechanism by using a pointer network similar to (See et al., 2017). This allows the decoder to predict words from the relevant regions as well as from the fixed vocabulary. At time step t, the decoder predicts the next word in the answer using the attention distribution, context vector and current word embedding. The attention distribution and context vector are obtained as follows:

$$o_i^t = v^T tanh(W_e e_i + W_h h_t + b_o), \quad (4)$$

$$\gamma^t = softmax(o_i^t), \quad (5)$$

	AW	AS/MS
document	659	28/66
question	10	n/a
answer	5	n/a

	NE
Train	32,747
Dev	3,461
Test	10,557

Table 1: NarrativeQA statistics. **AW, AS/MS** are defined as avg. words, avg./max. sentences in the document/question/answer. **NE**: Number of examples.

where h_t is hidden state of the decoder, v, W_e, W_h, b_o are learnable parameters. The γ^t represents a probability distribution over words of relevant regions e_i. The context vector is given by $c_t = \sum_i \gamma_i^t e_i$.

The probability distribution to predict word w_t from the fixed vocabulary (P_{fv}) is computed by passing state h_t and context vector c_t to a linear layer followed by a softmax function denoted as

$$P_{fv} = softmax(W_v(X_v[h_t, c_t] + b_p) + b_q). \quad (6)$$

To allow decoder to copy words from the encoder sequence, we compute a soft gate (P_{copy}), which helps the decoder to generate a word by sampling from the fixed vocabulary or by copying from a selected text regions (ψ_r). The soft gate is calculated as

$$P_{copy} = \sigma(w_p^T c_t + v_h^T h_t + w_x^T x_t + b_c), \quad (7)$$

where x_t is current word embedding, h_t is hidden state of the decoder, c_t is the context vector, and w_p, v_h, w_x, and b_c are learnable parameters. We maintain a list of out-of-vocabulary (OOV) words for each document. The fixed vocabulary along with this OOV list acts as an extended vocabulary for each document. The final probability distribution (unnormalized) over this extended vocabulary (P_{ev}) is given by

$$P_{ev}(w_t) = (1 - P_{copy})P_{fv}(w_t) + P_{copy} \sum_{i:w_i=w_t} \gamma_i^t. \quad (8)$$

2.4 Training

We jointly estimate the parameters of our model coming from the *Co-attention*, *Context Zoom*, and *Answer Generation* layers, which are denoted as θ_a, θ_z, and θ_g respectively. Estimating θ_a and θ_g is straight-forward by using the cross-entropy objective $J_1(\{\theta_a, \theta_g\})$ and the backpropagation algorithm. However, selecting text regions in the *Context Zoom* layer makes it difficult to estimate θ_z

given their discrete nature. We therefore formulate the estimation of θ_z as a reinforcement learning problem via a policy gradient method. Specifically, we design a reward function over θ_z.

We use mean F-score of ROUGE-1, ROUGE-2, and ROUGE-L (Lin and Hovy, 2003) as our reward function R. The objective function to maximize is the expected reward under the probability distribution of current text regions ψ_r, i.e., $J_2(\theta_z) = E_{p(r|\theta_z)}[R]$. We approximate the gradient $\nabla_{\theta_z} J_2(\theta_z)$ by following the REINFORCE (Williams, 1992) algorithm. To reduce the high variance in estimating $\nabla_{\theta_z} J_2(\theta_z)$ one widely used mechanism is to subtract a baseline value from the reward. It is shown that any number will reduce the variance (Williams, 1992; Zaremba and Sutskever, 2015), here we used the mean of the mini-batch reward b as our baseline. The final objective is to minimize the following equation:

$$J(\theta) = J_1(\{\theta_a, \theta_g\}) - J_2(\theta_z) + \sum_{i=1}^{B}(R^i - b), \quad (9)$$

where, B is the size of mini-batch, and R^i is the reward of example $i \in B$. $J(\theta)$ is now fully differentiable and we use backpropagation to estimate θ.

3 Experimental Results

3.1 Dataset

The NarrativeQA dataset (Kočiskỳ et al., 2017) consists of fictional stories gathered from books and movie scripts, where corresponding summaries and question-answer pairs are generated with the help of human experts and Wikipedia articles. The summaries in NarrativeQA are 4-5 times longer than documents in the SQuAD dataset. Moreover, answers are well-formed by human experts and are not verbatim in the story, thus making this dataset ideal for testing our model. The statistics of NarrativeQA are available in Table 1[1].

3.2 Baselines

We compare our model against reported models in Kočiskỳ et al. (2017) (Seq2Seq, ASR, BiDAF) and the Multi-range Reasoning Unit (MRU) in Tay et al. (2018). We implemented two baseline models (Baseline 1, Baseline 2) with *Context Zoom* layer similar to Wang et al. (2018). In both baselines we replace the span prediction layer with an answer generation layer. In Baseline 1 we use an attention based seq2seq layer without using copy mechanism in the answer generation unit similar to Choi et al. (2017). In Baseline 2 the answer generation unit is similar to our ConZNet architecture.

[1]please refer Kočiskỳ et al. (2017) for more details

3.3 Implementation Details

We split each document into sentences using the sentence tokenizer of the NLTK toolkit (Bird and Loper, 2004). Similarly, we further tokenize each sentence, corresponding question and answer using the word tokenizer of NLTK. The model is implemented using Python and Tensorflow (Abadi et al., 2015). All the weights of the model are initialized by Glorot Initialization (Glorot et al., 2011) and biases are initialized with zeros. We use a 300 dimensional word vectors from GloVe (Pennington et al., 2014) (with 840 billion pre-trained vectors) to initialize the word embeddings, which we kept constant during training. All the words that do not appear in Glove are initialized by sampling from a uniform random distribution between [-0.05, 0.05]. We apply dropout (Srivastava et al., 2014) between the layers with keep probability of 0.8 (i.e dropout=0.2). The number of hidden units are set to 100. We trained our model with the AdaDelta (Zeiler, 2012) optimizer for 50 epochs, an initial learning rate of 0.1, and a minibatch size of 32. The hyperparameter 'sample size' (number of relevant sentences) is chosen based on the model performance on the devset.

3.4 Results

Table 2 shows the performance of various models on NarrativeQA. It can be noted that our model with sample size 5 (choosing 5 relevant sentences) outperforms the best ROUGE-L score available so far by 12.62% compared to Tay et al. (2018). The low performance of Baseline 1 shows that the hybrid approach (ConZNet) for generating words from a fixed vocabulary as well as copying words from the document is better suited than span prediction models (Seq2Seq, ASR, BiDAF, MRU).

To validate the importance of finding relevant sentences in contrast to using an entire document for answer generation, we experimented with sample sizes beyond 5. The performance of our model gradually dropped from sample size 7 onwards. This result shows evidence that only a few relevant sentences are sufficient to answer a question.

We also experimented with various sample sizes to see the effect of intra sentence relations for an-

Model	BLEU-1	BLEU-4	ROUGE-L	METEOR
Seq2Seq [*]	15.89	1.26	13.15	4.08
ASR [*]	23.20	6.39	22.26	7.77
BiDAF [*]	33.72	15.53	36.30	15.38
MRU (Tay et al., 2018)	36.55	19.79	41.44	17.87
Baseline 1 (SS=5)	30.22	14.43	34.40	13.36
Baseline 2 (SS=5)	39.35	20.17	43.36	18.01
ConZNet (SS=1)	28.97	16.70	36.02	12.39
ConZNet (SS=3)	36.21	19.33	41.23	17.77
ConZNet (SS=5)	**42.76**	**22.49**	**46.67**	**19.24**
ConZNet (SS=7)	40.80	21.14	44.01	18.67

[*]These results are reported in Kočiskỳ et al. (2017)

Table 2: Performance of various models on NarrativeQA dataset (SS=sample size $\equiv$ number of relevant sentences)

swer generation. The performance of the model improved dramatically with sample sizes 3 and 5 compared to the sample size of 1. These results show that the importance of selecting multiple relevant sentences for generating an answer. In addition, the low performance of Baseline 2 indicates that just selecting multiple sentences is not enough, they should also be related to each other. This result points out that the self-attention mechanism in the *Context zoom* layer is an important component to identify related relevant sentences.

4 Conclusion

We have proposed a new neural-based architecture which condenses an original document to facilitate fast comprehension in order to generate better well-formed answers than span based prediction models. Our model achieved the best performance on the challenging NarrativeQA dataset. Future work can focus for example on designing an inexpensive preprocess layer, and other strategies for improved performance on answer generation.

References

Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. 2015. TensorFlow: Large-scale machine learning on heterogeneous systems. Software available from tensorflow.org.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. *In Proceedings of the International Conference on Learning Representations.*

Steven Bird and Edward Loper. 2004. Nltk: the natural language toolkit. In *Proceedings of the ACL 2004 on Interactive poster and demonstration sessions*, page 31. Association for Computational Linguistics.

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.

Eunsol Choi, Daniel Hewlett, Jakob Uszkoreit, Alexandre Lacoste, Illia Polosukhin, and Jonathan Berant. 2017. Coarse-to-fine question answering for long documents. In *Proceedings of the ACL*. Association for Computational Linguistics.

Christopher Clark and Matt Gardner. 2017. Simple and effective multi-paragraph reading comprehension. *CoRR*, abs/1710.10723.

Xavier Glorot, Antoine Bordes, and Yoshua Bengio. 2011. Deep sparse rectifier neural networks. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, volume 15 of *Proceedings of Machine Learning Research*, pages 315–323, Fort Lauderdale, FL, USA. PMLR.

Mandar Joshi, Eunsol Choi, Daniel Weld, and Luke Zettlemoyer. 2017. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1601–1611.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1746–1751.

Tomáš Kočiskỳ, Jonathan Schwarz, Phil Blunsom, Chris Dyer, Karl Moritz Hermann, Gábor Melis, and Edward Grefenstette. 2017. The narrativeqa reading comprehension challenge. *arXiv preprint arXiv:1712.07040*.

Chin-Yew Lin and Eduard Hovy. 2003. Automatic evaluation of summaries using n-gram co-occurrence statistics. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology - Volume 1*, NAACL '03, pages 71–78, Stroudsburg, PA, USA. Association for Computational Linguistics.

Michael E. J. Masson. 1983. Conceptual processing of text during skimming and rapid sequential reading. *Memory & Cognition*, 11(3):262–274.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392.

Abigail See, Peter J. Liu, and Christopher D. Manning. 2017. Get to the point: Summarization with pointer-generator networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1073–1083. Association for Computational Linguistics.

Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. 2017. Bidirectional attention flow for machine comprehension. *In Proceedings of the International Conference on Learning Representations*.

Yelong Shen, Po-Sen Huang, Jianfeng Gao, and Weizhu Chen. 2017. Reasonet: Learning to stop reading in machine comprehension. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1047–1055. ACM.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15:1929–1958.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112.

Yi Tay, Luu Anh Tuan, and Siu Cheung Hui. 2018. Multi-range reasoning for machine comprehension. *CoRR*, abs/1803.09074.

Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. 2015. Pointer networks.

Shuohang Wang, Mo Yu, Xiaoxiao Guo, Zhiguo Wang, Tim Klinger, Wei Zhang, Shiyu Chang, Gerald Tesauro, Bowen Zhou, and Jing Jiang. 2018. R^3: Reinforced reader-ranker for open-domain question answering.

Wenhui Wang, Nan Yang, Furu Wei, Baobao Chang, and Ming Zhou. 2017. Gated self-matching networks for reading comprehension and question answering. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 189–198.

Ronald J. Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Mach. Learn.*, 8(3-4):229–256.

Caiming Xiong, Victor Zhong, and Richard Socher. 2017. Dynamic coattention networks for question answering. *In Proceedings of the International Conference on Learning Representations*.

Wojciech Zaremba and Ilya Sutskever. 2015. Reinforcement learning neural turing machines. *CoRR*, abs/1505.00521.

Matthew D Zeiler. 2012. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*.