Exploring Efficacy of Embeddings on Relation Network for Natural Language Question Answering Task

Zhangsheng Lai[*], **Aik Beng Ng**[*]
Nvidia AI Technology Centre
NVIDIA
`{zlai,aikbengn}@nvidia`

Jin Xing Lim[*]
Engineering Systems Design
Singapore University of Technology and Design
`jinxing_lim@mymail.sutd.edu.sg`

1 Introduction

Deep learning has made it possible to do classification of objects in images and translation of languages, often with incredible accuracy. This is achieved due to the ability of neural networks to pick out important patterns that are inconceivable to the human eye, from large quantities of labeled data. However, just being able to learn patterns is not sufficient as it is not the only ability associated to intelligence; reasoning is another essential ability [1] that separates humans from machines. Hence, in recent years there is much work on reasoning related research, like visual reasoning [5, 11] where the machine is able to give an answer given an image and a visual question about the image, and text-based question answering [11] where the machine is able to answer a question based on the earlier sentences given to it.

In this project, we focus on the text-based question answering task using relation network (RN) [11] on the bAbI dataset [13]. RNs are networks that are designed based on relational reasoning, where its capacity to compute relations is baked into the architecture without having the neural network to learn it.

For any neural approach for natural language processing, word and sentence embeddings are indispensible. They allow us to represent words and sentences whose original forms are strings, as vectors which then we can feed it into an artificial neural network. There are various ways to embed words, and while unsupervised representations have been the more commonly used approach, using the assumption that you can tell a word by the company it keeps, there is an increased focus on supervised representations and also multi-task learning of representations. In our project, we explore how different types of embeddings, in particular, how the traditional unsupervised representations compare up to representations obtained from multi-task learning.

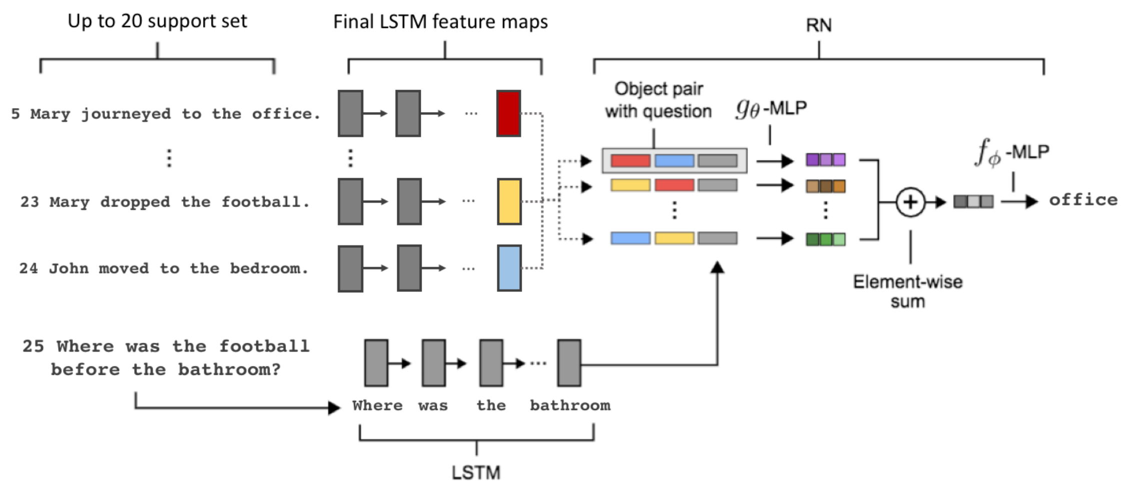
Our experiments involve using two different representations, the first one being the approach used by the original RN paper [11], to embed the context and questions into sentence embeddings using LSTMs, and the other uses the universal sentence encoder (USE) [3] to embed the context and questions into sentence embeddings. In the RN paper, sentence embeddings are called objects which the RN is used to learn the relation between them. We then compare the performance of different embeddings on RN for bAbI question answering task.

2 Task

2.1 bAbI

The bAbI dataset is a pure text-based question answering (QA) dataset that contains a total of 20 tasks. Each task corresponds to a particular type of reasoning, such as deduction, induction and counting. Every question is associated with a set of supporting facts, which provides the context for

[*] Indicates equal contribution.



Up to 20 support set
Final LSTM feature maps
RN
5 Mary journeyed to the office.
23 Mary dropped the football.
24 John moved to the bedroom.
25 Where was the football before the bathroom?
Object pair with question
g_θ-MLP
f_ϕ-MLP
office
Element-wise sum
Where was the bathroom
LSTM

Figure 1: Text based QA architecture. Contexts and questions are processed with an LSTM to produce a set of context and question embedding. Objects, colored yellow, red, and blue, are constructed using LSTMs or USE. The RN considers relations across all pairs of objects, conditioned on the question embedding, and integrates all these relations to answer the question. Our alternative approach substities the LSTM with the USE to produce the embeddings.

the question being asked. An example “Sandra picked up the football” and “Sandra went to the office” support the question “Where is the football?”, which we humans can arrive at an easy at the answer “office”. A task is considered to be successfully passed if it attains an accuracy of 95% or higher.

2.2 Two Supporting Fact Task

Task 2 of the bAbI tasks requires chaining of two or three supporting facts to answer the question. As such, to answer the question “Where is the football?”, it has to possibly be able to link information from the sentences “Mary moved to the bathroom”, “Mary picked up the football there” and “Mary went back to the garden” for it to conclude that the football is at the garden. This tasks makes it challenging for the neural network as it requires some form of memory for it to be able to link previously acquired knowledge to answer the question.

The RN succeeded on the basic induction task, which proved difficult for Sparse DNC, Differentiable Neural Computer (DNC) [4] and Sparse DNC [9]. However, it missed the 95% mark for the “two supporting fact” and “three supporting fact” tasks, which architectures with memory like Memory Networks [14], DNC and sparse DNC excelled. Thus we focused on task 2 for our project, to investigate if a better representation, in a form the USE obtained from multi-task learning, fare better than unsupervised representations.

3 Model

3.1 Relation Network

The RN is a neural network module which is designed to do relational reasoning. It is a composite function whose form is given by:

$$RN(O) = f_\phi\left(\sum_{i,j} g_\theta(o_i, o_j, q)\right) \tag{1}$$

where the input is a set of “objects” $O = \{o_1, \ldots, o_n\}$, $o_i \in \mathbb{R}^m$ is the i^{th} object, and f_ϕ and g_θ are function with learnable parameters ϕ and θ respectively. f_ϕ and g_θ are mulit-layer perceptrons (MLP), where g_θ learns the relation between any two given objects and f_ϕ maps the relations to one of the many possible answers. In the next part, we will discuss how using different embeddings lead to different implementations of our model.

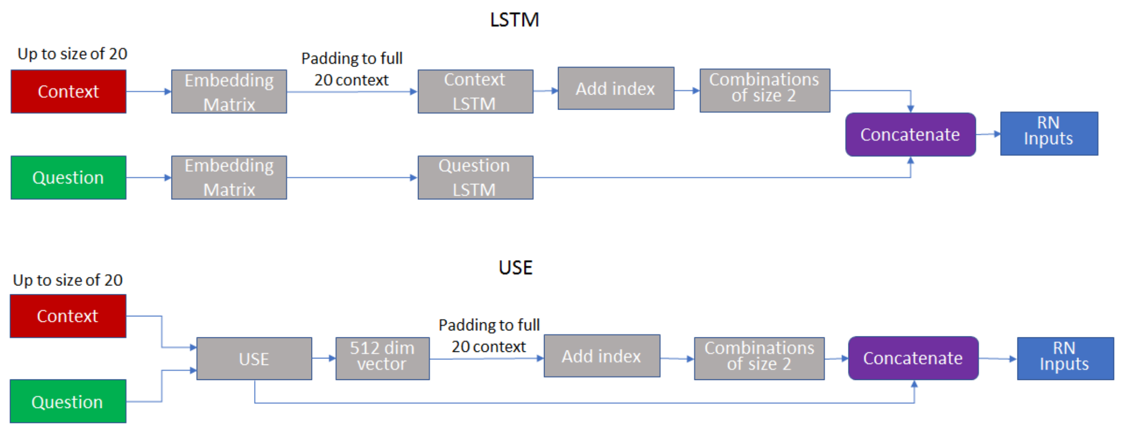


Figure 2: Embedding Pipelines. Padding are done at different stages of the pipline for the different embeddings in which padding is done first for the LSTM, whereas padding is done after passing through the USE.

3.2 Embeddings

There are various ways to embed words, from common unsupervised models like word2vec [6] and GloVe [7]. Early this year we have ELMo [8], which improved the state of the art embeddings. ELMo differs from other unsupervised embeddings by having their inputs to be characters instead of words, allowing them to take advantage of sub-word units to compute meaningful representations even for out-of-vocabulary words. Using ELMo representations to existing NLP systems have shown improvements to the state-of-the-art for tasks like SQuAD [10] and SNLI [2]. Thus we hypothesise that a better representation, which we are proposing to be USE, can give our bAbI task 2 a better result. We shall now discuss the two different embeddings we have used in our experiments whose efficacy we are investigating.

3.2.1 LSTM

For the LSTM embeddings, a randomly initialised word embedding matrix is used to assign each word to a vector representation, which is processes each sentence word-by-word through a 32 unit LSTM. A different word embedding matrix and LSTM is used to generate the context and question embedding respectively. The final state of the sentence-processing-LSTM is the object that is our input for the RN module.

LSTM Embeddings The bAbI suite of tasks requires transformation of the natural language inputs into a set of objects. In the original RN paper, to provide the prior knowledge for the RN to base on, 20 sentences in the support set that were prior to the question of interest were used as the context for answering the question, and each sentence was processed word-by-word with an 32 unit LSTM to obtain a sentence embedding, after which it is tagged with indexes to indicate their relative position in the support set to obtain the objects that are the inputs to our RN. A separate LSTM with 32 units was used to process the question.

From (1) we see that for a context of size k, there are $k(k-1)/2$ unique pairings of the objects. In its implementation, for questions with context sizes that are smaller than 20, we pad it with zeros such that we have a full context size of 20, then let the LSTM process the words to get the context embeddings. The indexs that tagged the relative position in the support set, which are just one-hot vectors indicating the position of the sentence in the support set, are then concatenated to the embeddings. Lastly, for each given question embedding, it is concatenated to all possible combination of size 2 to obtain the inputs to the RN, which has dimensions $(190, 136)$, the first value reflecting the number of possible combinations and the other the sum of the dimension of the objects and question embeddings.

3.2.2 USE

The universal sentence encoder (USE) provides sentence level embeddings that shows strong transfer performance on multiple NLP tasks. It also performs better than transfer learning using on word embeddings alone [3]. The USE is designed base on the observation that for a representation to be able to generalise over a range of diverse tasks, we have to encode multiple aspects of the sentence. Thus USE learns a universal sentence embedding by switching between several tasks.

There are two encoding models available; DAN is efficient but with lower accuracy, and transformer, has greater model complexity but is more computationally expensive. In our experiment, we use the transformer model, as a better representation of the sentence would help improve the question answering task. The transformer based sentence encoding constructs sentence embeddings using the transformer architecture from [12].

USE Embeddings For the USE embeddings, we followed the original RN paper of using up to 20 sentences in the support set as the prior for answering the question. The string form of the raw context and question sentences are fed into the USE which outputs a 512 dimensional vector sentence embedding for each sentence. As such we see that the USE embeddings gives the contexts and questions a much richer representation as compared to the LSTM embeddings.

One main difference from the LSTM embedding approach is that we do not pad the context to the full size of 20 before passing through the USE, compared to padding the context to 20 sentences before letting the LSTM process the words. We note that to do something similar in the USE context, we have to pad the sentences to the same length with a special character, for example "null". Instead we keep the sentence length as it is, pass it through the USE to obtain the sentence embeddings and for context with size less than 20, pad it sufficiently with zeros we get 20 contexts. The addition of the one-hot vector indexs, we only concatenate the one-hot vectors for those non-trivial contexts, for those padded contexts, the indexs are just zero vectors of dimension 20. From experiments, if we were to label the trivial contexts with the one-hot vectors, it gives a very poor performance. As the dimensions of the representation of each sentence for both context and questions is increased from 32 to 512, our inputs to the RN has dimensions $(190, 1536)$.

4 Experiments and Results

We have made several comparison between different types of embeddings, different sample sizes, different types of inputs to feed to the RN module and different modifications to the RN module with our implementation of USE embeddings. We compared the accuracy of the test set for different modifications to the RN module and compared the accuracy of the validation set for the rest of the comparisons.

For the experiments, all models were run with batch size of 64, with 200 epochs and validation set was taken from the last 10% of the training set. For the original RN module, $f_\phi\left(\sum_{i,j} g_\theta(o_i, o_j, q)\right)$, g_θ is a four-layer MLP 256 unites per layer and f_ϕ is a three-layer MLP consisting of 256, 512 and 6 units, where the final softmax output was optimized with a cross-entropy loss function using the Adam optimizer with a learning rate of $2e^{-4}$. All models are run on original RN module except for the different modifications to the RN module. The results and weights used are all based on the epoch that gave the best validation accuracy.

4.1 Comparison between Different Embeddings

We have made comparisons between the two different embeddings of the sentences of the contexts, questions and answers from all data from the training set of Task 2:

- **Model LSTM**: Use LSTM to embed the sentences of the contexts, questions and answers, and then to pass to the RN module

- **Model USE**: Use USE to embed the sentences of the contexts, questions and answers, and then to pass to the RN module

Model	Training Accuracy (%)	Validation Accuracy (%)
LSTM	-	35.0
USE	100	87.1

Table 1: Results between LSTM and USE Embeddings

From the results, we can see that based on training solely on the training set of Task 2, using USE embedding provides a much better representation than LSTM embedding in training our RN module. Thus, we ran the rest of the models with USE embedding.

4.2 Comparison between Different Sample Sizes

We ran the original model and compared between 2 different sample sizes of the training set:

- **First 1000** contexts of the training set
- **Full 9995** contexts of the training set

Model	Training Accuracy (%)	Validation Accuracy (%)
First 1000	96.2	35.0
Full 9995	100	87.1

Table 2: Results between first 1000 and full 9995 training data

Based on the validation accuracy, we can see that it is necessary to include all data in the training set for training of the models. Thus, we ran the rest of the models based on the full 9995 contexts.

4.3 Comparison between RN inputs with and without Indexes

We made comparison between different modifications of inputs before passing to the RN module:

- **RN inputs with no index**: Removed the index of every sentence of each context prior to each question
- **RN inputs with indexs**: Includes both the sentence embeddings and index of every sentence of each context prior to each question (as of what was implemented in the Santoro paper [11])

RN inputs	Training Accuracy (%)	Validation Accuracy (%)
Without Index	100	76.9
With Index	100	87.1

Table 3: Results between first 1000 and full 9995 training data

Based on the validation accuracies, it seems that the indexing of the sentences in the contexts prior to each question serves an important attribute for the RN module to train on. Thus, we included indexing for the RN inputs for rest of the models that would be implemented.

4.4 Comparison between Different Modifications of the RN Module

For LSTM embedding, each sentence and question had been embedded to a 32-unit object, with sentences from the contexts and questions being processed through separate LSTMs. For each question embedding, it is paired with a combination of 2 sentence embeddings from up to 20 prior sentences, with indexes. Batches of 190 (20 sentences choose 2) combinations of dimension 136 vectors (32-unit sentence i embedding + one hot 20-dimension vector for index of sentence i + 32-unit sentence j embedding + one hot 20-dimension vector for index of sentence j + 32-unit question embedding) with batch size of 64 were passed through the RN module.

On the other hand, each sentence and question had been embedded to a 512-unit object with the pre-trained USE. As a result, instead of having 190 combinations of dimension 136 vectors with batch size of 64 being passed through the RN module, we would have combinations of dimension 1576 (512-unit sentence i embedding + one hot 20-dimension vector for index of sentence i + 512-unit sentence j embedding + one hot 20-dimension vector for index of sentence j + 512-unit question embedding). From this we can see that there is an increase in the dimension of each input, and hence, the complexity of each input before passing through the RN module. Thus, we believed by increasing the number of layers or/and number of neurons in each layer would improve the accuracy of the test set as the RN module would have more components to learn from.

We compared between different modifications of the RN model and checked for the accuracy of the test set:

- **Model ori**: Original RN module ($g : [256 \times 4], f : [256, 512, 6]$)
- **Model g6**: Increase number of layers of g from 4 to 6 ($g : [256 \times 6], f : [256, 512, 6]$)
- **Model g8**: Increase number of layers of g from 4 to 8 ($g : [256 \times 8], f : [256, 512, 6]$)
- **Model 512**: Increase number of neurons in each layer of g from 256 to 512 ($g : [512 \times 4], f : [256, 512, 6]$)
- **Model 512f512**: Increase number of neurons in each layer of g from 256 to 512 and first layer of f from 256 to 512 ($g : [512 \times 4], f : [512, 512, 6]$)
- **Model g6x512**: Increase number of layers of g from 4 to 6 and increase number of neurons in each layer of g from 256 to 512 ($g : [512 \times 6], f : [256, 512, 6]$)
- **Model g6x512f512**: Increase number of layers of g from 4 to 6, increase number of neurons in each layer of g from 256 to 512 and first layer of f from 256 to 512 ($g : [512 \times 6], f : [512, 512, 6]$)
- **Model double**: Increase number of layers of g from 4 to 6 and double the number of neurons in all layers except for the last layer of f ($g : [512 \times 6], f : [512, 1024, 6]$)

Model	Training Accuracy (%)	Validation Accuracy (%)	Test Accuracy (%)
ori	100	87.1	85.5
g6	100	89.1	88.4
g8	18.4	22.2	20.0
512	100	88.2	85.0
512f512	99.9	87.9	88.3
g6x512	100	87.8	88.5
g6x512f512	99.7	88.3	89.0
double	100	88.2	88.4

Table 4: Results between different modifications of the RN module

Based on the test accuracies of the above models, we can see that increasing the number of layers of g function from 4 to 6 increased the test accuracy but drastically decreased the accuracy when it was increased to 8 layers. There is also improvement in the test accuracy when the number of neurons for each layer of g function from 256 to 512. The above table shows that the best model for our USE implementation is Model g6x512f512, which is to increase both the number of layers from 4 to 6 and the number of neurons from 256 to 512 for each layer of the g function, together with an increase of number of neurons from 256 to 512 for the first layer of the f function.

5 Discussion and Conclusions

This project showed us how different sentence embeddings can yield a huge difference in the performance in the RN by training on Task 2 of the bAbI dataset alone, which requires relational reasoning based on two supporting facts. For the original implementation using LSTM embeddings, it was implemented on all the training data of all 20 tasks, scoring above 95% for 18 out of 20 tasks. However, when it is implemented solely on the training set of Task 2, the validation accuracy drops significantly to 35% (93.5% when training with all 20 tasks' training data). Our experiments showed that using USE embeddings can obtain surprisingly good task performance (validation accuracy of 87.1%) with remarkably little task specific training data.

We also showed that using the full training set provides more information for the RN module to train on in order to yield a higher validation accuracy. Furthermore, it seems to have a positive correlation between the sample sizes and the validation accuracies (training from first 1000 contexts yielded 35.0%, while training from the full 9995 contexts yielded 87.1%). This provides greater motivation for us to modify our implementation to train on all training data of all 20 tasks as it will provide us with a much larger training set.

Our experiments also showed the importance of knowing the locations of sentences of the context prior to each question in training the RN. With the additional of the index of each sentence in our training, the validation accuracy increased from 76.9% (without index) to 87.1% (with index). Thus, relationship between the locations of the sentences prior to each question and the question itself was learned and is crucial in training the RN module.

As mentioned above, using USE would result in a higher dimensional vector for each object pair and question given. Due to this increase in dimension of our inputs to pass through the RN module, we believed that the complexity of the RN should increase as well for better training. We showed that by increasing the number of layers or/and the number of neurons in each layer in the g function to a certain extent would yield slightly better performance. By increasing the number of layers from 4 to 6, the validation accuracy has increased by 2.0%. Moreover, increasing the number of neurons in each layer from 256 to 512 increased the accuracy by 0.9%. Our experiments showed that the best model is the one which increased of the number of layers in the g function to 6, number of neurons in each layer of g function to 512 and first layer of the f function to 512 neurons (test accuracy of 89.0% as compare to the original model of 85.5%).

We would like to continue this work further even after submitting this project report. From the results of our experiments, we believe that by training on all the 20 tasks of the bAbI dataset, we would expect higher test accuracy on not only Task 2, but also improving the performance of all tasks. However, we could also foresee that improvement in performance for all tasks may not be as straightforward as we thought as each of the tasks requires different types of logical inference skills. Thus, training the RN module with different kind of tasks may seem to be a challenging task. Nonetheless, we wish to continue to use USE in the embeddings of the sentences and questions for all the training sets of the 20 tasks to see if we can succeed in more than 95% for all the tasks.

References

[1] L. Bottou. From Machine Learning to Machine Reasoning. *Arxiv preprint arXiv11021808*, page 15, 2011. ISSN 0885-6125. doi: 10.1007/s10994-013-5335-x. URL `http://arxiv.org/abs/1102.1808`.

[2] S. R. Bowman, G. Angeli, C. Potts, and C. D. Manning. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, 2015.

[3] D. Cer, Y. Yang, S.-y. Kong, N. Hua, N. Limtiaco, R. S. John, N. Constant, M. Guajardo-Cespedes, S. Yuan, C. Tar, Y.-H. Sung, B. Strope, and R. Kurzweil. Universal Sentence Encoder. 2018. ISSN 1042-1629. doi: 10.1007/s11423-014-9358-1. URL `http://arxiv.org/abs/1803.11175`.

[4] A. Graves, G. Wayne, M. Reynolds, T. Harley, I. Danihelka, A. Grabska-Barwińska, S. G. Colmenarejo, E. Grefenstette, T. Ramalho, J. Agapiou, A. P. Badia, K. M. Hermann, Y. Zwols, G. Ostrovski, A. Cain, H. King, C. Summerfield, P. Blunsom, K. Kavukcuoglu, and D. Hassabis. Hybrid computing using a neural network with dynamic external memory. *Nature*, 538(7626): 471–476, 2016. ISSN 14764687. doi: 10.1038/nature20101. URL `http://dx.doi.org/10.1038/nature20101`.

[5] J. Johnson, B. Hariharan, L. V. D. Maaten, J. Hoffman, L. Fei-Fei, C. L. Zitnick, and R. Girshick. Inferring and Executing Programs for Visual Reasoning. *Proceedings of the IEEE International Conference on Computer Vision*, 2017-Octob:3008–3017, 2017. ISSN 15505499. doi: 10.1109/ICCV.2017.325.

[6] T. Mikolov, K. Chen, G. Corrado, and J. Dean. Efficient estimation of word representations in vector space. *CoRR*, abs/1301.3781, 2013. URL `http://dblp.uni-trier.de/db/journals/corr/corr1301.html#abs-1301-3781`.

[7] J. Pennington, R. Socher, and C. D. Manning. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, 2014. URL `http://www.aclweb.org/anthology/D14-1162`.

[8] M. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237. Association for Computational Linguistics, 2018. doi: 10.18653/v1/N18-1202. URL `http://aclweb.org/anthology/N18-1202`.

[9] J. W. Rae, J. J. Hunt, T. Harley, I. Danihelka, A. Senior, G. Wayne, A. Graves, and T. P. Lillicrap. Scaling Memory-Augmented Neural Networks with Sparse Reads and Writes. (Nips), 2016. ISSN 10495258. doi: 10.1016/j.molimm.2007.07.004. URL `http://arxiv.org/abs/1610.09027`.

[10] P. Rajpurkar, J. Zhang, K. Lopyrev, and P. Liang. Squad: 100, 000+ questions for machine comprehension of text. *CoRR*, abs/1606.05250, 2016. URL `http://arxiv.org/abs/1606.05250`.

[11] A. Santoro, D. Raposo, D. G. T. Barrett, M. Malinowski, R. Pascanu, P. Battaglia, and T. Lillicrap. A simple neural network module for relational reasoning. pages 1–16, 2017. ISSN 21607516. doi: 10.1109/WACV.2017.108. URL `http://arxiv.org/abs/1706.01427`.

[12] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. u. Kaiser, and I. Polosukhin. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 5998–6008. Curran Associates, Inc., 2017. URL `http://papers.nips.cc/paper/7181-attention-is-all-you-need.pdf`.

[13] J. Weston, A. Bordes, S. Chopra, A. M. Rush, B. van Merriënboer, A. Joulin, and T. Mikolov. Towards AI-Complete Question Answering: A Set of Prerequisite Toy Tasks. 2015. ISSN 03787753. doi: 10.1016/j.jpowsour.2014.09.131. URL `http://arxiv.org/abs/1502.05698`.

[14] J. Weston, S. Chopra, and A. Bordes. Memory Networks. pages 1–15, 2015. ISSN 1098-7576. doi: v0. URL `https://arxiv.org/pdf/1410.3916.pdf`.