# ALICE: A Natural Language Question Answering System Using Dynamic Attention and Memory

Tushar Prakash(✉), Bala Krushna Tripathy, and K. Sharmila Banu

School of Computer Science and Engineering, VIT, Vellore, India
tusharprk@yahoo.com, {tripathybk,sharmilabanu.k}@vit.ac.in

**Abstract.** With the growing amount of textual information in recent years, it has become quite challenging to keep up with content produced by humans. Many models have been proposed that can perform reading comprehension on a variety of texts; however, past models either excel at information retrieval on complex texts or inference on simple texts. In this paper, we propose a model called ALICE that can perform information retrieval as well as inference tasks on any text. It is scalable to any document size and can be used to aid professionals in quickly finding answers to their problems using natural language queries. We will explore how ALICE achieves this and test it on some common datasets.

**Keywords:** Language · Processing · Reading · Comprehension · Attention Memory · Question · Answer

## 1 Introduction

In recent years, many models have been developed that can perform reading comprehension. Most recent models utilize deep learning techniques augmented with attention and memory mechanisms [1, 2] to decode answers from the encoded questions and documents [1, 3]. Many models either use information retrieval or inference techniques to find answers. However, current state-of-the-art models still face a decline in accuracy when processing larger and more complex text documents [3–6]. In this paper, we introduce a new approach towards text comprehension by utilizing predictive word embeddings, matched attention and external memory. Our proposed model can handle any document size due to its expandable architecture. Additionally, we predict the semantic relationship of out-of-vocabulary (OOV) and rare words, thereby, allowing us to generate significantly more accurate vector based word embeddings. End to end training is also possible with this model because the entire model is differentiable. The general architecture of this model can be expanded to applications beyond reading comprehension, such as, sentiment classification, image captioning and machine translation.

© Springer Nature Singapore Pte Ltd. 2018
I. Zelinka et al. (Eds.): ICSCS 2018, CCIS 837, pp. 274–282, 2018.
https://doi.org/10.1007/978-981-13-1936-5_30

## 2 Related Work

Most natural language processing models use recurrent neural networks (RNNs) for encoding and decoding operations as they can handle sequential data very easily. More recently, LSTM and GRU [7] model variations have been adopted because they can encode larger contexts [6, 7]. The recent addition of attention mechanisms over gated RNNs has also enabled a plethora of applications including, but not limited to, sentiment analysis, neural machine translation [1, 3], image caption generation [8] and question answering [2]. Furthermore, external memory manipulation also involves attention as seen in [4]. In this paper, we use a similar approach to understand the context of a document. However, unlike previous approaches [6, 9], our model combines the benefits of inference capability in augmented memory models with the scalability of sequential gated attention based models. This allows our model to perform a variety of tasks that were previously limited to only certain types of models. In recent years vector embedding models such as Word2Vec [10] and GloVe [11] have become a common method for encoding words because they can capture some of the semantic relationships between words. However, a major drawback of these models is that they can't generate representations of words or phrases that were not part of their training set. So, rare or out-of-vocabulary (OOV) words either have poor representations or don't have any representation at all. This results in diminished accuracy because the attention based models can't effectively determine how the poorly represented word relates to the context. Methods to mitigate this issue have been described in [12]. In this paper we propose a new method for embedding words that takes care of the underlying issues with vector based word embedding models, thereby significantly boosting their accuracy. To test our model we use common datasets like Facebook's bAbI dataset [13]; Stanford's SQuAD dataset [14] and Microsoft's MS MARCO dataset [15]. The bAbI dataset tests inference tasks over simple sentences, whereas, The SQuAD and MS MARCO datasets test more complex inference and information retrieval aspects of reading comprehension.

## 3 Model and Methods

In this section we provide a brief overview of the proposed model. Subsequently, we describe each component of the model in detail and give the intuition for its creation. We begin the task of reading comprehension by sequentially encoding the question and document using two bidirectional GRUs [7, 16]. A paired matching matrix then associates each word in the question with words in the given document [17]. The relevance of each sentence is determined using a soft attention mechanism on the matching matrix. Subsequently, a temporal controller writes the weighted encoding of the word into memory using a method similar to the one described in [18]. The encoded question and the word pair matching matrix is then passed as input to the read controller which selects the weighted encodings of the words (memory vector) that are relevant to the question.

Each selection is fed back to the read controller along with the question to find more evidence for supporting an answer (Fig. 1).

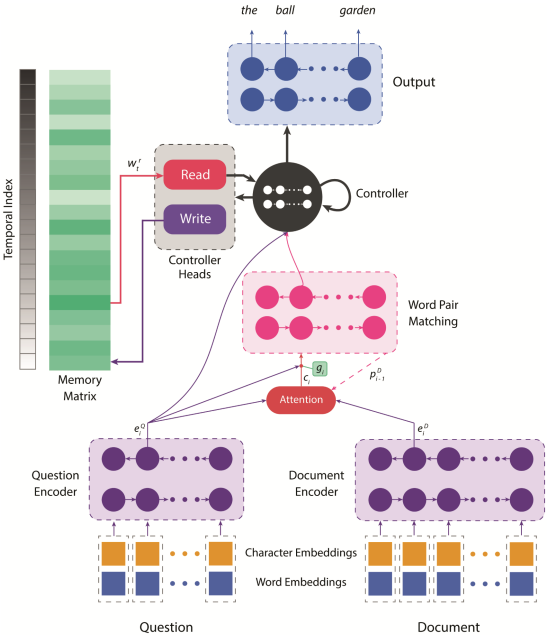


**Fig. 1.** A high level view of ALICE's architecture showing one time step

The weighted results are given to a bidirectional GRU which decodes the answer [1].

### 3.1 Embedding and Encoding

We propose a new method to handle rare or OOV words in vector based word embeddings [10, 11] by guessing the context of the words. A bidirectional LSTM (BiLSTM) is trained separately to predict the next word in a sentence. For each rare or OOV word, the BiLSTM generates 'n' candidate results to replace that word. We average the 'n' candidate vectors and insert then result into our word embedding model. We train the BiLSTM on texts used to train the word embedding model to ensure we don't encounter any OOV words. To encode the question Q and the document D, we generate their corresponding word embeddings $\left[w_i^Q\right]_{i=1}^m$ and $\left[w_i^D\right]_{i=1}^k$ along with their character embeddings $\left[c_i^Q\right]_{i=1}^m$ and $\left[c_i^D\right]_{i=1}^m$, where 'm' is the number of words in the question Q and 'k' is the number of words in the document D. For word embedding we use pre-trained GloVe embeddings [11] and for character embedding we use an LSTM-charCNN [19]. We then combine the word and character embeddings using bidirectional GRUs [16] to form encodings $\left[e_i^Q\right]_{i=1}^m$ and $\left[e_i^D\right]_{i=1}^k$ for all words in the question and document respectively.

$$e_i^Q = BiGRU\left(e_{i-1}^Q, \left[w_i^Q, c_i^Q\right]\right) \tag{1}$$

$$e_i^D = BiGRU\left(e_{i-1}^D, \left[w_i^D, c_i^D\right]\right) \tag{2}$$

By combining the enhanced word vector representations with character embeddings we get an encoding that effectively handles rare and OOV words.

### 3.2 Word to Word Relevance

To determine the importance of a word in answering the given question, we follow the suggestions outlined in [17] to generate word pair representations. For a given question encoding $\left[e_i^Q\right]_{i=1}^m$ and a document encoding $\left[e_i^D\right]_{i=1}^k$, the word pair representation $\left[p_i^D\right]_{i=1}^k$ is calculated using soft-alignment of words:

$$p_i^D = RNN\left(p_{i-1}^D, c_i\right) \tag{3}$$

where $c_i$ is a context vector formed by merging all word pair attention vectors with question's encoding $e_i^Q$.

$$c_i = \sum_{i=1}^{m} a_i e_i^Q \tag{4}$$

$$s_i = \frac{\exp\left(a_j\right)}{\sum_{j=1}^{m} \exp\left(a_j\right)} \tag{5}$$

$$a_i = w^T tanh\left(W^Q e_i^Q + W^D e_i^D + W^p p_{i-1}^D\right) \tag{6}$$

Here, $a_i$ is an attention vector over the individual question-document word pairs and $s_i$ is the softmax over $a_i$. w is a learned weight vector parameter whose transpose is $w^T$. In [20], they add $e_i^D$ as another input to the recurrent network used in $\left[p_i^D\right]$ (Fig. 2):

$$p_i^D = RNN\left(p_{i-1}^D, \left[e_i^D, c_i\right]\right) \tag{7}$$

Since the document may be very large, we introduce a gate $g_i$ over the input $\left[e_i^D, c_i\right]$. This allows us to find parts of the document that are relevant to the question.

$$g_i = sigmoid\left(W_g\left[e_i^D, c_i\right]\right) \tag{8}$$

$$\left[e_i^D, c_i\right]' = g_i \odot \left[e_i^D, c_i\right] \tag{9}$$

The gate $g_i$ is a learned value over time. This gate filters out the irrelevant words when $g_i$ is closer to zero and gives importance to words when $g_i$ is closer to one. The gate $g_i$ learns different values for $W_g$ over various time steps. Thus modeling a mechanism to effectively select parts of the document relevant to the question. In our model we use a BiGRU in place of an RNN.

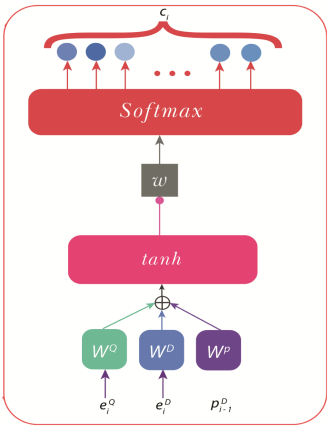


**Fig. 2.** A visual depiction of the attention mechanism used

### 3.3 Memory Controller

The memory architecture is similar to, but less complex than the one described in [18]. Memory is stored in an N × M matrix where N is the number of memory locations and M is the vector size at each location. Our first step is to write the weighted word vectors in to memory. This is achieved by using an LSTM for the controller network defined by:

$$i_t^l = sigmoid\left(W_i^l\left[x_t, h_{t-1}^l, h_t^l - 1 + b_i^l\right]\right) \tag{10}$$

$$f_t^l = sigmoid\left(W_f^l\left[x_t, h_{t-1}^l, h_t^l - 1 + b_f^l\right]\right) \tag{11}$$

$$s_t^l = f_t^l s_{t-1}^l + i_t^l tanh\left(W_s^l\left[x_t, h_{t-1}^l, h_t^l - 1 + b_s^l\right]\right) \tag{12}$$

$$o_t^l = sigmoid\left(W_o^l\left[x_t, h_{t-1}^l, h_t^l - 1 + b_o^l\right]\right) \tag{13}$$

$$h_t^l = o_t^l tanh\left(s_t^l\right) \tag{14}$$

where $l$ denotes the layer of the LSTM and sigmoid is the logistic sigmoid function defined as:

$$sigmoid(x) = \frac{1}{1 + e^{-x}} \tag{15}$$

$i_t^l, f_t^l, s_t^l, o_t^l$, and $h_t^l$ are the input, forget, state, output and hidden gates, respectively, at layer l and time t. The input vector $x_t$ is supplied to the controller at each time-step t. Since we want to maintain the order of sentences occurring in the document, we concatenate an increasing time index i to $p_i^D$ from the word matching step:

$$p_i^D = RNN\left(p_{i-1}^D, \left[e_i^D, c_i\right]\right) \tag{16}$$

$$x_t = \left[p_i^D, i\right] \tag{17}$$

To read the memory vectors, we use the weighted averages across all locations:

$$r_t^i = M_t^T w_t^r \tag{18}$$

Here, $M_t$ denotes the memory matrix at time $t$. The read vectors are appended to the controller input after each time-step. For the first time step, the question's encoding $e_Q^i$ is supplied as input. The read weighting $w_t^r$ determines the importance of a vector for answering the question. It is defined as:

$$w_t^r = f_t[1](i-1) + f_t[2]C(M,k) + f_t[3](i+1) \tag{19}$$

Here, $f_t$ is a read mode decided by applying a softmax over a collection of 3 states: move backward, find similar vectors, move forward. The read weight $w_t^r$ is applied over all memory locations, allowing the model to select important facts relevant to the given question. To find evidence supporting an answer, we use content-based addressing [18] on the read head to perform lookups over the memory:

$$C(M,k) = \frac{\exp\left(D(k, M[i,\cdot])\right)}{\sum_j \exp\left(D\left(k, M\left[j,\cdot\right]\right)\right)} \tag{20}$$

Here, $k \in R$ is the key or address of a memory location and D is the cosine similarity function. In our case, the temporal index is also used as the key.

### 3.4 Output

The output vectors are fed into a BiGRU which decodes the answer. Once the output is decoded, we calculate the loss using a standard cross-entropy loss function. We use this to minimize the sum of log probabilities when comparing the decoded answer with the actual one:

$$L(y,z) = -\sum_{\{i=0\}}^{m} z \cdot \log\left(P(y|z)\right) \tag{21}$$

Here, m is the vocabulary size, z is the actual answer and y is the prediction given by the model. The error is calculated and back-propagated until it is minimized.

## 4 Results

The bAbI dataset [13] consists of 20 tasks for testing a model's ability to reason over text. From the results listed in Table 1, we observe that ALICE performs significantly better than the DNC on basic induction tasks (task 16), which significantly contributes

to a higher mean accuracy for ALICE. We also observe that the DMN [4] performs better than ALICE on basic induction tasks, yet, ALICE performs better on all other tasks. The accuracy of ALICE converges towards 96.8% over 10 training sessions. We only report the best results obtained from one of the 10 sessions.

**Table 1.** Comparison of results obtained on the bAbI dataset

| Task | DMN [4] | DNC [18] | ALICE |
|---|---|---|---|
| 1. Single supporting fact | **100** | **100** | **100** |
| 2. Two supporting facts | 98.2 | **99.6** | 98.6 |
| 3. Three supporting facts | 95.2 | **98.2** | 97.2 |
| 4. Two argument relations | **100** | **100** | 99.9 |
| 5. Three argument relations | 99.3 | **99.5** | 99.4 |
| 6. Yes/no questions | **100** | **100** | **100** |
| 7. Counting | 96.9 | **99.8** | 99.7 |
| 8. List/sets | 96.5 | **99.9** | 99.4 |
| 9. Simple negation | **100** | **100** | **100** |
| 10. Indefinite knowledge | 97.5 | **99.8** | 98.1 |
| 11. Basic co-reference | 99.9 | **100** | **100** |
| 12. Conjunction | **100** | **100** | **100** |
| 13. Compound co-reference | 99.8 | **100** | **100** |
| 14. Time reasoning | **100** | 99.7 | 99.8 |
| 15. Basic deduction | 100 | 100 | 100 |
| 16. Basic induction | **99.4** | 47.6 | 73.44 |
| 17. Positional reasoning | 59.6 | **88** | 82.6 |
| 18. Size reasoning | 95.3 | **99.2** | 97.5 |
| 19. Path finding | 34.5 | **99.9** | 98.3 |
| 20. Agent motivations | 100 | 100 | 100 |
| Average accuracy | 93.605 | 96.56 | **97.209** |
| Number tasks failed (accuracy < 95%) | 2 | 2 | 2 |

We further test our model on the SQuAD [14] and MS MARCO datasets [15]. The SQuAD dataset [14] contains question-answer pairs derived from 536 Wikipedia articles. SQuAD uses exact match (EM) and F1 score metrics to measure the performance of a given model. We report the best results obtained from 1 of 10 training sessions in Table 2.

**Table 2.** A comparison of results obtained on the SquAD dataset

| Model | EM | F1 |
|---|---|---|
| SAN ensemble model [23] | **79.608** | **86.496** |
| Reinforced mnemonic reader ensemble model [22] | 77.7 | 84.9 |
| ReasoNet ensemble model [23] | 73.4 | 81.8 |
| ALICE | 78.024 | 85.212 |

While ALICE outperforms competitive models like ReasoNet [23] and Reinforced Mnemonic Reader [22]; the Stochastic Attention Network (SAN ensemble model) [21] still beats ALICE by a small margin. We attribute this to SAN's ensemble nature. To test our model (ALICE) on larger texts we use the MS MARCO dataset [15]. It contains multiple passages extracted from anonymized Bing search engine queries and the answers may not be exactly worded in those passages. The metrics used for evaluating a model for the MS MARCO dataset are BLEU and ROUGE-L scores.

From the results in Table 3, we see that ReasoNet [23] marginally outperforms ALICE on the MS MARCO dataset. Note that the results for ReasoNet are obtained by Microsoft AI and Research group after the paper was published. From our tests, we can clearly see that ALICE performs similar to, or better than some competitive models on the bAbI [13], SQuAD [14] and MS MARCO [15] datasets.

**Table 3.** A comparison of results obtained on the SquAD dataset

| Model | BLEU | ROUGE-L |
|---|---|---|
| ReasoNet [23] [Microsoft AI and research results] | **38.81** | **39.86** |
| ALICE | 38.43 | 38.67 |

## 5 Conclusion

In this paper, we propose a novel model, ALICE, aimed at the task of reading comprehension and question answering. We simplify an existing state-of-the-art architecture and combine it with a matching layer, to attend over a question and document. We also provide a method to improve the accuracy of similar models by using a bidirectional LSTM to generate contextual word embeddings for out-of-vocabulary words. Results for our model show that it is scalable in size and complexity. Our model achieves results that are close to the state-of-the-art and similar to, or better than some competitive models. Future work includes simplifying the current model and applying this model to generate captions for images.

## References

1. Bahdanau, D., Cho, K., Bengio, Y.: Neural machine translation by jointly learning to align and translate (2014). arXiv preprint arXiv:1409.0473
2. Sukhbaatar, S., Weston, J., Fergus, R., et al.: End-to-end memory networks. In: Advances in Neural Information Processing Systems, pp. 2440–2448 (2015)
3. Luong, M.T., Pham, H., Manning, C.D.: Effective approaches to attention-based neural machine translation (2015). arXiv preprint arXiv:1508.04025
4. Kumar, A., et al.: Ask me anything: dynamic memory networks for natural language processing. In: International Conference on Machine Learning, pp. 1378–1387 (2016)
5. Gong, Y., Bowman, S.R.: Ruminating reader: reasoning with gated multi-hop attention (2017). arXiv preprint arXiv:1704.07415
6. Sordoni, A., Bachman, P., Trischler, A., Bengio, Y.: Iterative alternating neural attention for machine reading (2016). arXiv preprint arXiv:1606.02245

7. Chung, J., Gulcehre, C., Cho, K., Bengio, Y.: Empirical evaluation of gated recurrent neural networks on sequence modeling (2014). arXiv preprint arXiv:1412.3555
8. Vinyals, O., Toshev, A., Bengio, S., Erhan, D.: Show and tell: a neural image caption generator. In: 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 3156–3164. IEEE (2015)
9. Iyyer, M., Boyd-Graber, J., Claudino, L., Socher, R., Daumé III, H.: A neural network for factoid question answering over paragraphs. In: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pp. 633–644 (2014)
10. Mikolov, T., Sutskever, I., Chen, K., Corrado, G.S., Dean, J.: Distributed representations of words and phrases and their compositionality. In: Advances in Neural Information Processing Systems, pp. 3111–3119 (2013)
11. Pennington, J., Socher, R., Manning, C.: Glove: global vectors for word representation. In: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pp. 1532–1543 (2014)
12. Luong, M.-T., Sutskever, I., Le, Q.V., Vinyals, O., Zaremba, W.: Addressing the rare word problem in neural machine translation (2014). arXiv preprint arXiv:1410.8206
13. Weston, J., et al.: Towards AI-complete question answering: a set of prerequisite toy tasks (2015). arXiv preprint arXiv:1502.05698
14. Rajpurkar, P., Zhang, J., Lopyrev, K., Liang, P.: Squad: 100,000 + questions for machine comprehension of text (2016). arXiv preprint arXiv:1606.05250
15. Nguyen, T., et al.: MS MARCO: a human generated machine reading comprehension dataset (2016). arXiv preprint arXiv:1611.09268
16. Cho, K., et al.: Learning phrase representations using RNN encoder–decoder for statistical machine translation. In: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pp. 1724–1734 (2014)
17. Rocktäschel, T., Grefenstette, E., Hermann, K.M., Kočiský, T., Blunsom, P.: Reasoning about entailment with neural attention (2015). arXiv preprint arXiv:1509.06664
18. Graves, A., et al.: Hybrid computing using a neural network with dynamic external memory. Nature **538**(7626), 471 (2016)
19. Kim, Y., Jernite, Y., Sontag, D., Rush, A.M.: Character-aware neural language models. In: AAAI, pp. 2741–2749 (2016)
20. Wang, S., Jiang, J.: Learning natural language inference with LSTM. In: Proceedings of NAACL-HLT, pp. 1442–1451 (2016)
21. Liu, X., Shen, Y., Duh, K., Gao, J.: Stochastic answer networks for machine reading comprehension (2017). arXiv preprint arXiv:1712.03556
22. Hu, M., Peng, Y., Qiu, X.: Mnemonic reader for machine comprehension (2017). CoRR, abs/1705.02798 http://arxiv.org/abs/1705.02798
23. Shen, Y., Huang, P.-S., Gao, J., Chen, W.: ReasoNet: learning to stop reading in machine comprehension. In: Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 1047–1055. ACM (2017)