Learning to Transform, Combine, and Reason in Open-domain Question Answering

Dehghani, M.; Azarbonyad, H.; Kamps, J.; de Rijke, M.

DOI
10.1145/3289600.3291012

Publication date
2019

Document Version
Final published version

Published in
WSDM'19

Learning to Transform, Combine, and Reason in Open-Domain Question Answering

Mostafa Dehghani
University of Amsterdam
dehghani@uva.nl

Hosein Azarbonyad
University of Amsterdam
h.azarbonyad@uva.nl

Jaap Kamps
University of Amsterdam
kamps@uva.nl

Maarten de Rijke
University of Amsterdam
derijke@uva.nl

ABSTRACT

Users seek direct answers to complex questions from large open-domain knowledge sources like the Web. Open-domain question answering has become a critical task to be solved for building systems that help address users' complex information needs. Most open-domain question answering systems use a search engine to retrieve a set of candidate documents, select one or a few of them as context, and then apply reading comprehension models to extract answers. Some questions, however, require taking a broader context into account, e.g., by considering low-ranked documents that are not immediately relevant, combining information from multiple documents, and reasoning over multiple facts from these documents to infer the answer. In this paper, we propose a model based on the Transformer architecture that is able to efficiently operate over a larger set of candidate documents by effectively combining the evidence from these documents during multiple steps of reasoning, while it is robust against noise from low-ranked non-relevant documents included in the set. We use our proposed model, called TraCRNet, on two public open-domain question answering datasets, SearchQA and Quasar-T, and achieve results that meet or exceed the state-of-the-art.

CCS CONCEPTS

• **Information systems** → **Question answering**; *Retrieval models and ranking*; • **Computing methodologies** → *Neural networks*;

KEYWORDS

Question answering; Multihop reasoning; Transformer; Universal transformer.

ACM Reference Format:
Mostafa Dehghani, Hosein Azarbonyad, Jaap Kamps, and Maarten de Rijke. 2019. Learning to Transform, Combine, and Reason in Open-Domain Question Answering. In *The Twelfth ACM International Conference on Web Search and Data Mining (WSDM '19), February 11–15, 2019, Melbourne, VIC, Australia.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3289600.3291012

WSDM '19, February 11–15, 2019, Melbourne, VIC, Australia

ACM ISBN 978-1-4503-5940-5/19/02…$15.00
https://doi.org/10.1145/3289600.3291012

1 INTRODUCTION

Open-domain question answering aims to satisfy users who are looking for a direct answer to a complex information need. This requires querying large open-domain knowledge sources like the Web. Inferring the answer to a question given multiple documents that potentially contain the answer, is at the heart of the open-domain question answering task. Most open-domain question answering systems described in the literature first retrieve relevant documents or passages, select one or a few of them as the context, and then feed the question and the context to a reading comprehension system to extract the answer [3, 4, 13, 29]. However, information needed to answer complex questions is not always contained in a single, directly relevant document that is ranked high. In many cases, there is a need to read multiple documents, combine them, and reason over the facts from these documents to be able to give the correct answer to the question.

For example, in Figure 1, in order to infer the correct answer to the question: "`Who is the Spanish artist, sculptor and draughtsman famous for co-founding the Cubist movement?`" given the top-ranked document, a reading comprehension system most likely will extract "`Georges Braque`" as the answer, which is not the correct answer. In this example, in order to infer the correct answer, one has to go down the ranked list, gather and encode facts, even those that are not immediately relevant to the question, like "*Malaga is a city in Spain*," which can be inferred from a document at rank 66, and then in a multi-step reasoning process, infer some new facts, including "*Picasso was a Spanish artist*" given documents at ranks 12 and 66, and "*Picasso, who was a Spanish artist, co-founded the Cubist*" given the previously inferred fact and the document ranked third.

In this example, and in general in many cases in open-domain question answering, a piece of information in a low-ranked document that is not immediately relevant to the question, may be useful to fill in the blanks and complete information extracted from the top relevant documents and eventually support inferring the correct answer. However, most open-domain question answering methods focus on only one or a few candidate documents by filtering out the less relevant documents to avoid dealing with noisy information and operate over the selected set of documents to extract the answer [26, 34, 35].

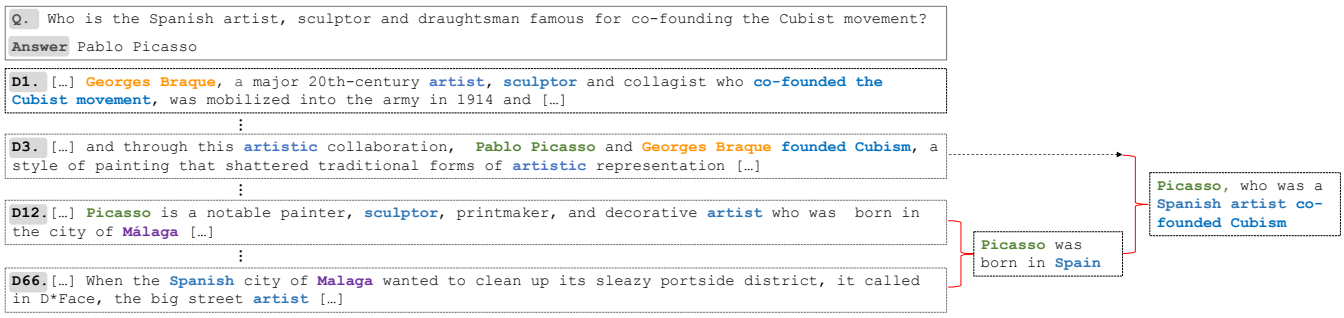


Figure 1: Example complex question answering that requires that information from multiple documents be combined and some amount of reasoning over the information extracted from those documents. (Best viewed in color.)

In this paper, we propose TraCRNet (pronounced *Tracker Net*) to improve open-domain question answering by explicitly operating on a larger set of candidate documents during the whole question answering process and learning how to aggregate and reason over information from these documents in an effective way while trying not to be distracted by noisy documents. Given the candidate documents and the question, to generate the answer, TraCRNet first **Tra**nsforms them into vectors by applying a stack of Transformer [32] blocks with self-attention over words in each document in a layer called *Input Encoding*. Then, it updates the learned representations from the first stage by **C**ombining and enriching them through a multihop **R**easoning process by applying multiple steps of the Universal Transformer [11] in a layer called *Multihop Reasoning*.

Returning to the example in Figure 1, after learning representations for each top-ranked document and the question, TraCRNet updates them by applying multiple steps of the Universal Transformer. Given the self-attention mechanism and inductive bias of the Universal Transformer, in the first step, TraCRNet can update the representation of document D#12 by attending to D#66 (as they are related by both mentioning Malaga) and augment the information in D#12 with the fact that "Malaga is city in Spain," so the updated vector of D#12 has the fact that "Picasso is a Spanish artist" encoded in itself. Then, in the next step of reasoning, TraCRNet can update the representation of D#3 by attending over the vector representing D#12 estimated in the previous step, and enrich the information in D#3 with the fact that "Picasso is a Spanish artist," and the updated vector of D#3 has the fact that "Picasso, who was a Spanish artist co-founded Cubism" encoded in it. After that, during answer generation, the decoder can attend to the final vector representing D#3 and give the correct answer.

TraCRNet has a number of desirable features. First, all the building blocks of TraCRNet are based on self-attentive feed-forward neural networks, hence per-symbol hidden state transformations are fully parallelizable, which leads to an enormous speedup during training and a super fast input encoding during inference time compared to RNN based models. Second, while there is no recurrence in time in our model, the recurrence in depth in the Universal Transformer used in the *Multihop Reasoning* layer, adds the inductive bias to the model that is needed to go beyond understanding each document separately and combine their information in multiple steps. Third, TraCRNet has the global receptive field of the Transformer based models [11, 32], which helps it to better encode a long document during *Input Encoding* as well as perform better inference over a rather large set of documents during *Multihop Reasoning*. And fourth, the hierarchical usage of a self-attention mechanism, first over words and then over documents, helps TraCRNet to control its attention both at word and document levels, making it less fragile to noisy input, which is of key importance while encoding many documents. All these properties of TraCRNet come together and lead to an effective and efficient architecture for open-domain question answering.

We employ TraCRNet on two public open-domain question answering datasets, SearchQA and Quasar-T, and achieve results that meet or exceed the state-of-the-art.

2 RELATED WORK

Open-domain question answering aims at answering a user's question using open and available external sources [17]. Different external knowledge sources such as individual textual documents [6] and collections of documents [33] have been used for answering questions in open-domain question answering settings.

With the advent of publicly available datasets such as the Stanford QA dataset [28] and the TREC question answering collections [33], the task of question answering has attracted a lot of attention. Here, the task is given a question and a passage, to extract the answer to the question. Neural network based models [25, 27, 36, 43] are the most successful approaches in this area. The overall idea behind most of these models is chunking the passage (locating the boundary where the answer lies) and extracting the answer. Although these approaches are related to ours, the main difference is that in our setting instead of one passage, we are given a set of documents from which the answer should be extracted.

2.1 Machine reading comprehension

Machine reading comprehension based models [4, 10, 13, 22, 24, 34] are the most used methods for extracting an answer in the question answering task. These models mostly use an attention mechanism to find the most relevant parts of the given documents to the question and try to combine these parts and extract the answer to the question. Most of the existing systems for open-domain QA rely on a search&read approach in which first a retrieval system

is used for extracting potentially relevant documents (passages) and then a model is used to infer the answer from the retrieved documents [4, 8, 14, 15, 21, 34, 35].

Chen et al. [4] are the first who attempt to use the search&read approach and use a reading comprehension system for reading and comprehending. It is argued that retrieving documents based on simple similarity metrics (TF-IDF) can result in a noisy set of documents (partially relevant documents) and this can have a negative effect on the performance of the QA system [8, 34]. To overcome this problem, Choi et al. [8] and Wang et al. [34] try to divide the question answering part into paragraph selection and answer selection steps in which first the most relevant paragraphs are determined and then an answer is extracted from them.

To avoid neglecting useful information contained in the paragraphs that are not selected in the paragraph selection step, Wang et al. [35] weight the paragraphs based on the expected information included in them and aggregate the information extracted from different paragraphs.

A similar approach is used by Lin et al. [26] in which first a paragraph selector is used to filter out noisy paragraphs and then a paragraph reader is employed to extract essential information from paragraphs. Finally, all information extracted from different paragraphs is aggregated to form the answer. The main focus of this approach is on rapidly skimming all available paragraphs and focusing more on the selected paragraphs.

We also try to aggregate information included in documents to answer open-domain questions. However, instead of RNNs as in Wang et al. [35], we use the Universal Transformer and the recurrence in depth for multihop reasoning and add inductive bias to the model to be able to capture all the complementary information in several documents. Moreover, instead of filtering out documents, we let the model attend to all documents and extract information from them.

2.2 Multihop reasoning

Multihop reasoning is one of the key requirements for reading comprehension [13, 30, 40]. The main intuition behind multihop reasoning is extracting essential information needed to answer a question in multiple steps. Our method also extracts answers of questions in multiple steps, however, our setting is different as our task is open-domain questions answering and the input to our task is a ranked list of relatively long documents.

Using the Universal Transformer [11] as a multihop reasoning layer in our model allows us to have a notion of *temporal states*, similar to the idea of dynamic memory networks [23], and our model updates its states (memory) in each step based on the output of previous steps, and this chain of updates can be viewed as steps in a multihop reasoning process. It is worth mentioning that the idea of multi-step inference has previously been used in other tasks such as document summarization [7] and classification [41].

Most of the previous approaches focused on removing noise in the retrieval steps of the question answering pipeline [4, 14, 15, 19, 34]. However, in our proposed model, we assume that the retrieval step is fixed and the main challenge is to generate an answer for the question given a set of documents that can potentially contain the answer or relevant facts that support understanding the question.

3 TRACRNET

In the setup we consider here, the model is given a question q and a set of n relevant documents $C_q = \{D_1^q, D_2^q, \dots D_n^q\}$ retrieved from the web using a search engine as the input, and the goal is to "generate" the answer a_q to the question q based on the supporting document(s) in the set C_q.

This is different from the standard Reading Comprehension (RC) tasks [18, 39]. First of all, in RC a single document (passage) is given, from which the answer should be extracted. Secondly, in RC a strong supervision on the positions of the answer spans is available during training. We also assume that the utilized information or techniques to retrieve relevant documents are not available to the model, therefore there is no leverage for getting better-supporting documents.

In this paper, we propose TraCRNet, which adapts a Transformer based architecture [11, 32] to the open-domain question answering setup. TraCRNet is based on the encoder-decoder architecture, where we have a hierarchy of transformer-based models in the encoder, where the model can attend first over words and then over documents [12]. At the bottom, in the *Input Encoding* layer, we encode each document in C_q as well as the question with transformer blocks with tied parameters that are fed by word-level embeddings. Then, we feed the encoded documents and the question from this layer to the *Multihop Reasoning* layer which is, in fact, a universal transformer block where representations of all documents and the question get iteratively updated using multiple steps of self-attention. Then, we use a stack of transformer decoder blocks as the *Output Decoder* layer to generate the answer.

The general schema of TraCRNet is depicted in Figure 2. Below, we explain the details of each of these layers in the model.

3.1 Input encoding

The Input Encoding layer is in charge of encoding each of the documents and the question to single vectors given their words' embeddings. To do so, given matrix $H_0 \in \mathbb{R}^{m \times d}$, where m is the number of tokens in the document/question and each row is a d-dimensional embedding of the token at each position of the sequence, we add a positional embedding PE to encode a notion of the order in the sequence to the embedding of each token. PE can be learned during training, but similar to [32], we compute the sinusoidal position embedding vectors for the position p separately for each dimension v:

$$\begin{aligned} PEp, 2v &= \sin(p/10,000^{2v/d}) \\ PEp, 2v+1 &= \cos(p/10,000^{2v/d}). \end{aligned} \tag{1}$$

After adding PE to H_0, we transform this matrix through a stack of N Transformer Encoder [32] blocks. In each block we compute a representation H_i for all m positions in parallel by applying the multiheaded dot-product self-attention mechanism, followed by a feed forward network function on H_{i-1}. We wrap each of these functions by residual connections and apply dropout [31] and layer normalization [1] (see the `Transformer Encoder` in Figure 2.).

In the attention function, we use the scaled dot-product attention:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d}}\right)V, \tag{2}$$

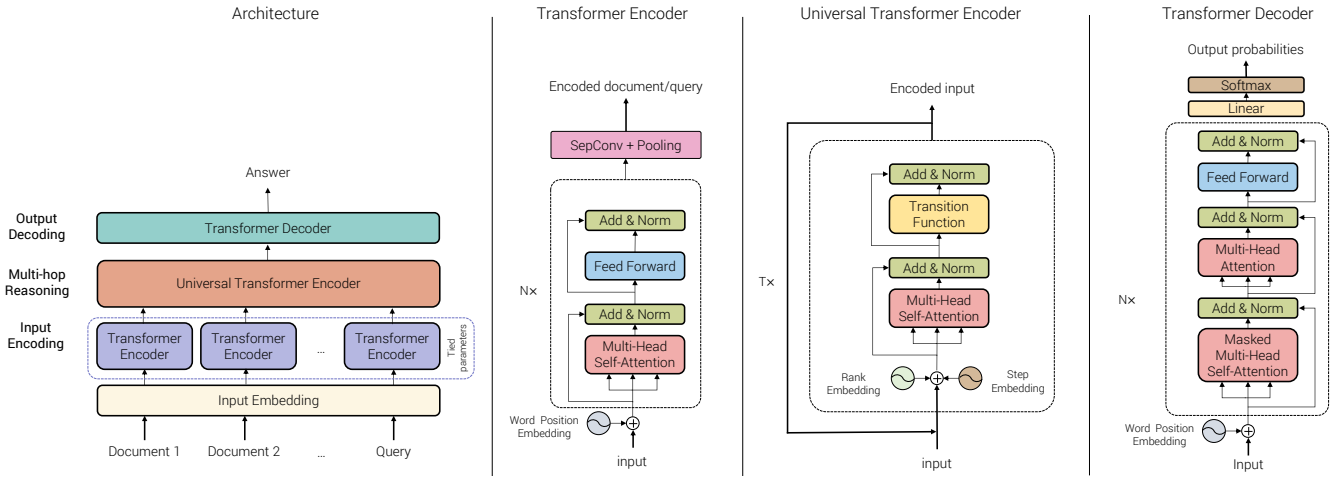


Figure 2: An overview of the TraCRNet architecture.

where Q, K, and V, denote attention query, attention key, and attention value matrices, respectively. We use multi-head attention with k heads, as introduced in [32],

$$\text{MultiHeadSelfAttention}(H) = \text{Concat}(\text{head}_1, \ldots, \text{head}_k)W^O, \quad (3)$$

where

$$\text{head}_j = \text{Attention}(HW_j^Q, HW_j^K, HW_j^V),$$

with different projections using trainable parameter matrices $W^Q \in \mathbb{R}^{d\times d/k}$, $W^K \in \mathbb{R}^{d\times d/k}$, $W^V \in \mathbb{R}^{d\times d/k}$ and $W^O \in \mathbb{R}^{d\times d}$, which is a linear projection. So in each block, we update the representation of the input tokens as follows:

$$H_i = \text{LayerNorm}(A_{i-1} + \text{FFN}(A_i)), \quad (4)$$

where

$$A_i = \text{LayerNorm}(H_{i-1} + \text{MultiHeadSelfAttention}(H_{i-1})) \quad (5)$$

and FFN$(\cdot)$ is a fully connected feed-forward network, which is applied to each position separately and identically:

$$\text{FFN}(x) = \max(0, x.W_1 + b_1)W_2 + b_2. \quad (6)$$

This stack of N Transformer Encoder blocks is followed by a depth-wise separable convolution [9, 20] and then a pooling function to get a single vector representation for the whole document (or the question). Depth-wise separable convolution is defined by a convolution on each of the feature channels separately, followed by a point-wise convolution that is applied to project them to a feature vector with the desirable depth (see [9] for more details).

3.2 Multihop reasoning

Multihop Reasoning is the layer in which the Universal Transformer [11] is employed to combine evidence from all documents with respect to the question within a multi-step process with the capacity of multihop reasoning. The Universal Transformer is an extension of the Transformer that has a strong inductive bias by introducing recurrency in depth. It iteratively refines the representation for all the input vectors in different positions over T steps. In our model, the input of the Universal Transformer Encoder is the set of vectors each representing a document in C_q or the question, that are computed by the *Input Encoding* layer.

In each step of the Universal Transformer, given $H_t \in \mathbb{R}^{(|C_q|+1)\times d}$ and the dimension d of the input vectors, we add two embeddings to H^t: a *Rank Embedding* that encodes the rank of documents given by the retrieval system (also used to distinguish the question from documents) and a *Step Embedding* that shows the model how many recurrent steps in depth have been taken so far. We compute both of these embeddings at time-step t as a single matrix, RSE^t, where its elements are computed as follows:

$$\begin{aligned} RSE^t_{r,2v} &= \sin(r/10,000^{2v/d}) \oplus \sin(t/10,000^{2v/d}) \\ RSE^t_{r,2v+1} &= \cos(r/10,000^{2v/d}) \oplus \cos(t/10,000^{2v/d}), \end{aligned} \quad (7)$$

where $\oplus$ is the elementwise summation, r is the rank of the document if the input vector represents a document, and $r = 0$ if the input vector represents the question. Following [11], we then apply multi-head self-attention and the transition function, where each of these modules is wrapped by residual connections with dropout [31] and layer normalization [1] on top of them (see the `Universal Transformer Encoder` in Figure 2):

$$H^t = \text{LayerNorm}(A^t + \text{Transition}(A^t)), \quad (8)$$

where

$$\begin{aligned} A^t = \text{LayerNorm}((H^{t-1} + RSE^t) + \\ \text{MultiHeadSelfAttention}(H^{t-1} + RSE^t)). \end{aligned} \quad (9)$$

In our experiments, we use depthwise separable convolution [9] as the Transition$(\cdot)$ function. The MultiheadSelfAttention$(\cdot)$ function we used here is described in Equation 3.

In the multihop reasoning layer, the representations of all the documents and question learned from the previous layer get updated during T steps of iterating over the Universal Transformer Encode block. Self-attention in this layer allows the model to understand each of the documents based on the information in all the as well as the question. In addition, the depth-wise recurrency in the Universal Transformer establishes connections among documents

at each step and lays the ground for performing multihop reasoning to solve cases similar to what we have shown in Example 1.

3.3 Output decoder

After T steps of refining the representations of documents and the question in the Universal Transformer Encoder, the final output is a matrix of d-dimensional vector representations $H \in \mathbb{R}^{(|C_q|+1)\times d}$ for all the documents in C_q and the question q.

Given this, we use a stack of Transformer Decoder blocks that share the basic architecture of the Transformer Encoder blocks used in the *Input Encoding*. However, first of all, masking is applied to the self-attention function to prevent attending to the feature tokens during decoding. Second, after the self-attention function, there is an Encoder-Decoder attention (which is equivalent to the normal attention introduced in [2]), where the decoder attends to the output of the *Multihop Reasoning* layer, i.e., H, using the same multi-head dot-product attention function from Equation 3, but with Q obtained from projecting the Transformer Decoder representations, and K and V obtained from projecting the Universal Transformer Encoder representations.

The output of the stack of Transformer Decoders is passed through a linear projection to transform from final decoder state to the output vocabulary size. Then a softmax is applied to get the per-token target distributions, yielding a $(m \times V)$-dimensional output matrix that is normalized over its rows.

To generate answer from the model at inference, we run the model autoregresively [16], where the model consumes the previously generated symbols at each time step in order to generate the distribution over the vocabulary for the next symbol. From this distribution, we select the symbol with highest probability as the next symbol.

3.4 Architectural choices

Transformer-based models have been shown to achieve impressive results on many sequence modeling tasks [5, 11, 32, 42] and they are easily parallelizable, making them an attractive alternative for RNNs. Besides their strength at sequence modeling and parallelizability, there are properties associated with them that are particularly helpful in our setup.

In TraCRNet, we use a Transformer [32] at the *Input Encoding* layer that receives token-level embeddings of documents/question and learns a single vector representation for them. In this level, first of all, dealing with long documents/passages is desirable and the *global receptive field* of the transformer model helps them to encode long sequences. In addition, although generalization is of key importance in all machine learning-based approaches, having a model with large capacity can help the model to *memorize* the meaning of infrequent words and improve the quality of the answers. We can simply train a relatively large transformer model in a really efficient time.

In the *Multihop Reasoning* layer, we use a Universal Transformer [11], which not only has the ability to generalize due to recurrence in depth but also possesses a strong inductive bias that enables the model to learn iterative or recursive transformations. On top of this, the Universal Transformer is Turing complete. These two properties can be crucial for tasks in which learning multi-step reasoning is needed. In the *Multihop Reasoning* layer, we already have representations of all documents and the question that are learned independently and a relatively light Universal Transformer model can do a great job in combining the information in different documents and reason about them, over multiple steps.

From an even broader point of view, the combination of the Transformer to encode local information and the Universal Transformer to combine global information will bring the model enough memorization as well as the ability of generalization.

We use depth-wise separable convolution on top of the Transformer model in the *Input Encoding* layer and as the transition function in the Universal Transformer in *Multihop Reasoning* layer. The main reason is that depthwise separable convolutions reduce the number of parameters and computation used in convolutional operations while increasing representational efficiency. We observed a better performance compared to the case of using a fully connected feed-forward network instead of depthwise separable convolution while it has a lower number of trainable parameters.

Table 1: Statistics of the datasets

Dataset	#train	#dev	#test
SearchQA	99,811	13,893	27,247
Quasar-T	28,496	3,000	3,000

4 EXPERIMENTAL SETUP

4.1 Datasets

We have conducted experiments on two publicly available open-domain question answering datasets: SearchQA [15] and Quasar-T [14]. In both of these datasets, candidate documents (passages) for each question have already been retrieved using a search engine and we do not add any extra documents to these result sets. On both datasets, the human performance is evaluated in a setup where the human subjects try to find the answers to the given question from the same documents retrieved by the IR model.

4.1.1 SearchQA. SearchQA[1] is a dataset of 140k question-answer pairs crawled from J! Archive, and augmented with text snippets retrieved using the Google search engine. For each question-answer pair, on average, about 50 web page snippets have been collected. In our experiments, we do not use the additional meta-data in the dataset like the snippet's URL.

4.1.2 Quasar-T. Quasar-T[2] consists of 43k open-domain trivia questions and their answers obtained from various internet sources. The set of candidate documents for each question is retrieved using "Lucene" from the ClueWeb09 corpus as the background corpus. In this dataset, for each question-answer pair, a set of 100 unique passages were collected as candidate documents.

4.2 Model configuration and experimental setup

We use WordPiece embeddings [38] with a 32k token vocabulary. In both *Input Encoder* and *Output Decoder* layers, we use a stack of 6 Transformer blocks with hidden_size = 512, num_attention_heads = 8, and batch_size = 2, 048. The rest of the hyper-parameters are set to the default values of the Transformer model. In the *Multihop*

[1] https://github.com/nyu-dl/SearchQA
[2] https://github.com/bdhingra/quasar

Reasoning layer, we have a Universal Transformer Encoder with hidden_size = 512 and num_attention_heads = 4. We set the number of recurrent steps in depth to 12. The rest of the hyper-parameters are set to the default values of the Universal Transformer model. We train with batch size of 4,096 tokens. We use Adam with learning rate of 1×10^{-9}, $\beta_1 = 0.9$, $\beta_2 = 0.98$, L_2 weight decay of 1×10^{-4}, learning rate warmup over the first 16,000 steps, and linear decay of the learning rate. We use a dropout probability of 0.1 on all layers. Since in our model answers are generated using the decoder instead of extracting from the context (detecting spam), to improve the quality of generation, we pretrain all the parameters of the Transformer decoder downstream of the task of language modeling. The embeddings are shared between encoder and decoder, thus the *Input Embedding* layer also enjoys the pretraining. This helps to improve the performance especially in terms of metrics that consider the exact match of the generated answer with the ground truth. During the training of the model, we use teacher-forcing, i.e., the decoder input is the gold target, shifted to the right by one position which is the usual setup for training autoregressive models [37].

In our experiments, TraCRNet and its variants are trained on 8 P100 GPUs for $800k$ training steps. For both datasets, a prepared version by Wang et al. [34] is used in our experiments to train and evaluate the TraCRNet as well as all the baselines. The statistics of the datasets are presented in Table 1. As the C_q, we consider top-50 top documents for the SearchQA, and top-100 for the Quasar-T. Following previous work on reading comprehension and open-domain question answering [3, 26, 30, 34, 35] as our evaluation metrics we adopt the F1 score, that loosely measures the average overlap between the predicted answer and the ground truth answer, and Exact Match (EM) that measures the percentage of predictions that match one of the ground truth answers exactly.[3]

5 RESULTS AND DISCUSSION

5.1 Baselines

We compare our results with the best reading comprehension and open-domain question answering models as well as research that achieves state-of-the-art on the SearchQA and Quasar-T datasets. To have a true apples-to-apples comparison, we only consider baselines that use no additional resources to solve the task for these datasets. We use the following methods as baselines:

(1) BiDAF [29], which is a reading comprehension model with bi-directional attention flow network that uses the concatenation of top-ranked candidate documents as the context.
(2) R^3 [34], which is a reinforcement learning approach that uses a ranker for selecting the most confident paragraph to train the reading comprehension model.
(3) Wang et al. [35]'s model, which learns to re-rank the answers extracted by applying an R^3 model on multiple documents based on coverage and strength of each of the documents given the question.
(4) Lin et al. [26]'s model, which is the most recent paper achieving state-of-the-art performance on the datasets we use for evaluation. They propose to decompose the process into a document selection to filter out noisy paragraphs, and a paragraph reader to extract the correct answer from the filtered documents. Finally, they aggregate multiple answers to obtain the final answer.

[3]We use the tool from SQuAD [28] for evaluation.

Table 2: Performance of TraCRNet compared to the baseline models

model	SearchQA		Quasar-T	
	EM	F1	EM	F1
BiDAF [29]	28.6	34.6	25.9	28.5
R^3 [34]	49.0	55.3	35.3	41.7
Wang et al. [35]	57.0	63.2	42.3	49.6
Lin et al. [26]	**58.8**	64.5	42.2	49.3
TraCRNet	52.9	**65.1**	**43.2**	**54.0**
Human Performance	43.9	–	51.5	60.6

Table 2 presents the results of the baseline models, TraCRNet, and the human performance on both datasets.

5.2 Main results

TraCRNet outperforms all the baselines and achieves a new state-of-the-art (to the best of our knowledge) on the Quasar-T dataset and performs as good as the best performing baseline on the SearchQA dataset. The main advantage of TraCRNet over the baselines is that it makes "full" use of the information of "all" the candidate documents in C_q. The models proposed by Lin et al. [26] and Wang et al. [35] are the strongest baselines on these datasets. Although they try to capture evidence from multiple sources by reranking or aggregating answers extracted from different documents, they filter out documents that are less likely to help at the beginning of the process. In this fashion, they lose the chance of using information from documents that are not directly relevant, like documents #12 or/and #66 in Example 1. However, TraCRNet keeps operating on the full set of candidate documents during the whole process and learns to which extent each document contributes to infer the final answer.

In SearchQA, we notice that for most of the questions, the answer can be extracted given a single document and in many cases, no multi-document multihop reasoning is required. Therefore, since TraCRNet *generates* the answer, as opposed to the baseline models that *extract* the answer from context, it gets a lower EM score. However, in terms of F1 score, TraCRNet slightly improves over the best baseline.

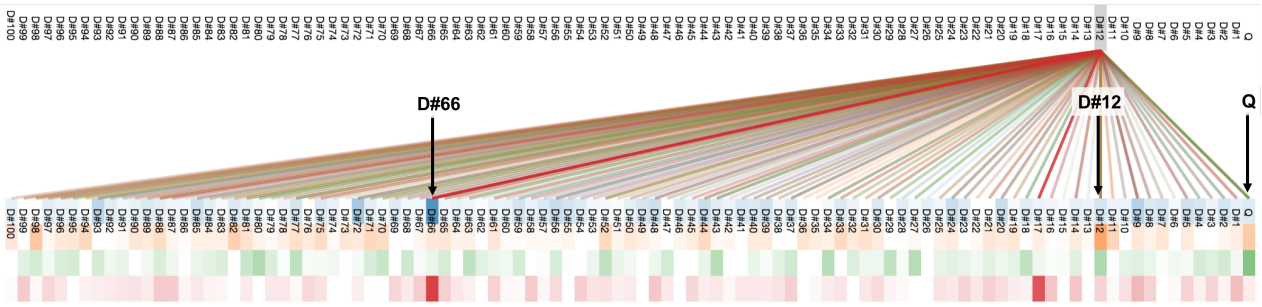
5.3 Effect of multihop reasoning

In order to investigate the effect of the *Multihop Reasoning* layer, we handicap TraCRNet by removing this layer and evaluate it in two cases:

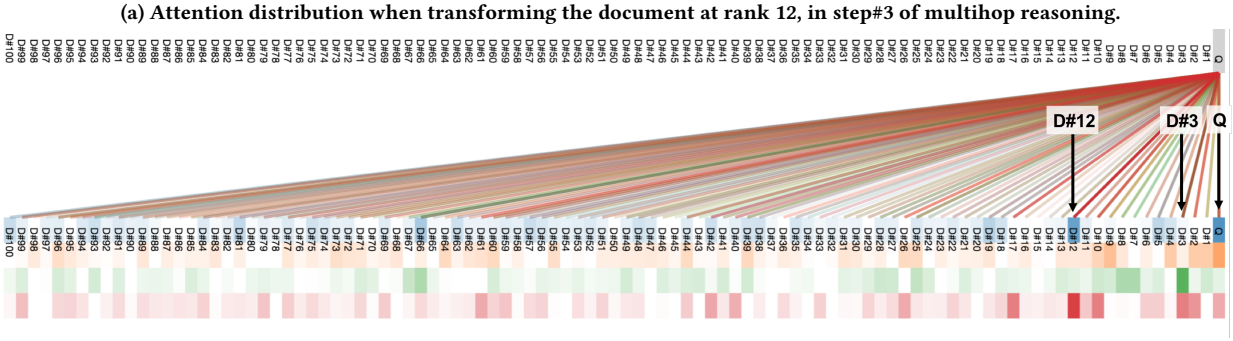
(1) $\text{TraCRNet}^{\text{d}}_{\text{no-mhr}}$, in which the decoder has access to document-level representations from the encoder, and
(2) $\text{TraCRNet}^{\text{w}}_{\text{no-mhr}}$ where pooling operation is removed and the decoder has access to word-level representations from the encoder.

Table 3 presents the results of the model in these situations.

On all measures and datasets, the performance drops when we remove the *Multihop Reasoning* layer. The drop in the performance is larger on the Quasar-T dataset than on the SearchQA dataset. We



(a) Attention distribution when transforming the document at rank 12, in step#3 of multihop reasoning.



(b) Attention distribution when transforming the question, in step#7 of multihop reasoning.

Figure 3: Visualization of multi-head self-attention on Multihop Reasoning layer of TraCRNet. (Best viewed in color.)

Table 3: Performance of TraCRNet with and without the *Multihop Reasoning* layer; numbers in parenthesis indicate percentage of performance loss

model	SearchQA		Quasar-T	
	EM	F1	EM	F1
TraCRNet	52.9	65.1	43.2	54.0
$\text{TraCRNet}^{\text{d}}_{\text{no-mhr}}$	48.6 (−8%)	61.7 (−5%)	36.4 (−16%)	43.6 (−19%)
$\text{TraCRNet}^{\text{w}}_{\text{no-mhr}}$	50.2 (−5%)	59.3 (−9%)	38.1 (−12%)	40.2 (−25%)

noticed that trivia questions in Quasar-T, in many cases, contain clauses that should be considered together with and/or operations to be able to give the correct answer. For instance, to answer the question "What Australian food was discovered by John McAdam," we should consider that "the food is Australian" *and* "the food is discovered by John McAdam." In this situation, the chance of having multiple documents each containing one of these facts increases. Thus, having multiple supporting documents and the need for reasoning (similar to Example 1) will be the exact point where the advantage of the *Multihop Reasoning* layer kicks in.

Another observation here is that when we remove the *Multihop Reasoning* layer, passing word-level embeddings from the encoder to the decoder leads to better EM scores, but not to improved F1 scores. The main reason is that, in this situation, access to the input words from the decoder is more explicit. This helps the model to get closer to answer extraction than pure answer generation.

For the test example that is presented in Figure 1, we observed that all baseline models output "Georges Braque" which is extracted from the document at rank 1. However, unlike all the baselines, TraCRNet returns the correct answer. We looked into the attention distributions in the *Multihop Reasoning* layer of TraCRNet at different steps (of the employed Universal Transformer with 12 depth-wise recurrent steps). We were able to find a relation between attention distributions and the reasoning steps that are needed to give the correct answer to this question. We illustrate this in Figure 3.

Figure 3a presents the attention distribution over all documents and the question while encoding the document at rank 12 at step 3. TraCRNet has a high level of attention for the document at rank 66 using heads 1 and 4 (blue and red) as well as for the question using head 3 (green) while transforming the document at rank 12. This is in accordance with the fact that the model first needs to update the information encoded in the document at rank 12 with the fact that "Malaga is a city in Spain" from the document at rank 66. Later, at step 7, while encoding the question (Figure 3b), TraCRNet attends over document 12, which has information about "Picasso who is a Spanish artist" (updated in step 3) using heads 1 and 4 and document 3, which contains information about "Picasso as a co-founder of Cubism" using head 2 (green).

5.4 Impact of the number of documents

As we explained before, unlike most of the previous work that filters candidate documents and narrows down the set of documents under consideration to either a single document or a small set of highly relevant documents before applying an answer extractor to them, TraCRNet uses the full set of candidate documents retrieved by the search engine during the entire process of generating the answer. This is of great advantage as our analysis shows that, for

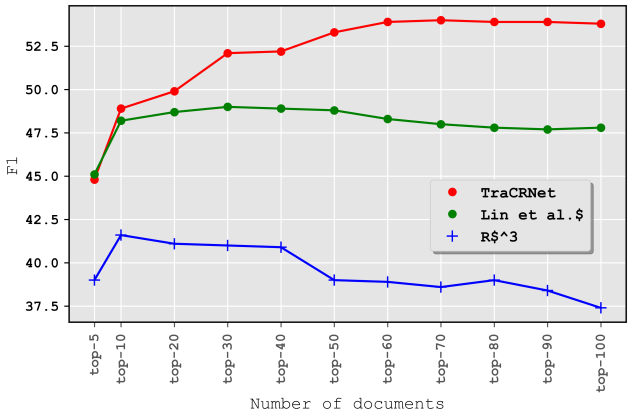


Figure 4: Performance in terms of F1 of TraCRNet and baselines (R^3 [34] and Lin et al. [26]'s model) with different numbers of candidate documents on Quasar-T dataset.

some questions, the correct answer can only be extracted when considering information from low-ranked documents that are not immediately relevant to the question. However, this can potentially come at the cost of (1) efficiency, as we need to process a larger input, and of (2) performance, as there will be more noisy and non-relevant documents when we go down the ranked list of candidate documents. Making use of self-attentive feed-forward neural networks as building blocks of TraCRNet brings the ability of full per-symbol parallelization and leads to an enormous speedup on encoding the input documents. This lets the model encode a larger set of candidate documents efficiently.

To study how the performance of TraCRNet is affected by the number of candidate documents, we train and evaluate TraCRNet as well as R^3 [34] and Lin et al. [26]'s model on the Quasar-T dataset, using different numbers of candidate documents associated with each question.[4] Figure 4 presents the performance of these models when they are fed with the top-5, top-10, ..., top-100 retrieved documents. As can be seen, although Lin et al. [26]'s model is pretty good at staying robust when noise increases (it is designed to learn from distant supervision), increasing the number of candidate documents eventually leads to a small drop in performance of both baselines due to the noise in the low-ranked documents. However, TraCRNet not only controls the effect of noisy low-ranked documents by calibrating their effect on inferring the final answer through self-attention, but it also keeps improving as we increase the number of documents as it can exploit any useful information contained in low-ranked documents which can help better understand the question or perform reasoning.

6 CONCLUSIONS

We have proposed TraCRNet, a method for inferring answers to questions in an open-domain question answering setting. TraCRNet is built around the Transformer to both encode long sequences of words and extract the answer from multiple documents which are potentially relevant to the question. TraCRNet empowers the encoder-decoder architecture used for reading comprehension with (1) a powerful multihop reasoning step which can aggregate the information inferred from multiple documents and the question, and (2) the inductive bias of the Universal Transformers which lets the model reason over multiple documents during multiple steps and go beyond understanding single documents. We have shown that TraCRNet has a stronger reasoning power than previous approaches for open-domain question answering and can outperform the state-of-the-art on two publicly available datasets for this task, i.e., SearchQA and Quasar-T.

In general, having more supporting documents associated with questions helps to find high-quality answers [35]. However, adding more documents can lower the performance due to the noise introduced by non-relevant documents. Hence, an effective open-domain question answering system should have a mechanism to handle noise effectively. Our analysis shows that, for some questions, the answer can only be extracted using information from low-ranked documents. However, most of the existing approaches focus on high-ranked documents and either do not consider the documents at lower ranks or even if they consider low-ranked documents, they do not have an effective mechanism to remove noise introduced by them. We showed that TraCRNet can both consider documents at lower ranks and remove noise when necessary. Besides that, multihop reasoning has a very high impact on the performance in the open-domain question answering task. This indicates that to achieve a good performance, it is very important to reason over multiple documents and have an effective approach to aggregate the evidence inferred from them to form the answers of questions, and TraCRNet excels at this.

There are multiple possible directions to extend the work done in this paper. In this research, we assumed that a set of documents from which the answer for a question should be extracted is available and given. The retrieval step can have a great impact on the performance. Here, in this paper, we viewed the retrieval step as a black box and focused on extracting the answer from given documents. In the end, by exploiting the available documents as much as possible TraCRNet outperformed the existing baselines that try to optimize both retrieval and reasoning steps [26, 34, 35]. To achieve even more improvements, it would be interesting to extend TraCRNet to jointly optimize both retrieval and reasoning steps.

Furthermore, we limited TraCRNet's decoder to attend over hidden states of the encoder at the document-level. An interesting line of future work would be to also let the model have access to the hidden states of the encoder at the token-level during decoding. This can improve the quality of generated answers as the decoder will have more explicit access to the individual tokens in the input.

Code

The code for re-running all of the experiments in the paper is available at https://github.com/MostafaDehghani/TraCRNet

Acknowledgments

This research was supported in part by the Netherlands Organization for Scientific Research through the *Exploratory Political Search* project (ExPoSe, NWO CI # 314.99.108), by the Digging into Data Challenge through the *Digging Into Linked Parliamentary Data* project (DiLiPaD, NWO Digging into Data # 600.006.014).

[4] In this experiment, we just change the initial number of candidates, but we train baseline models with their original setups and do not impose any assumption (e.g., fixing the candidate list) on them.

This research was also supported by Ahold Delhaize, the Bloomberg Research Grant program, Elsevier, the Google Faculty Research Awards program, the Innovation Center for Artificial Intelligence (ICAI), the Netherlands Institute for Sound and Vision, and the Netherlands Organization for Scientific Research (NWO) under project nrs CI-14-25, 652.002.001, 612.001.551, 652.001.003,

All content represents the opinion of the authors, which is not necessarily shared or endorsed by their respective employers and/or sponsors.

REFERENCES

[1] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. 2016. Layer Normalization. *arXiv preprint arXiv:1607.06450* (2016).
[2] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural Machine Translation by Jointly Learning to Align and Translate. In *International Conference on Learning Representations*.
[3] Christian Buck, Jannis Bulian, Massimiliano Ciaramita, Wojciech Gajewski, Andrea Gesmundo, Neil Houlsby, and Wei Wang. 2018. Ask the Right Questions: Active Question Reformulation with Reinforcement Learning. In *International Conference on Learning Representations*.
[4] Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. 2017. Reading Wikipedia to Answer Open-domain Questions. In *55th Annual Meeting of the Association for Computational Linguistics*. 1870–1879.
[5] Mia Xu Chen, Orhan Firat, Ankur Bapna, Melvin Johnson, Wolfgang Macherey, George Foster, Llion Jones, Niki Parmar, Mike Schuster, Zhifeng Chen, and others. 2018. The Best of Both Worlds: Combining Recent Advances in Neural Machine Translation. In *56th Annual Meeting of the Association for Computational Linguistics*. 76–86.
[6] Tongfei Chen and Benjamin Van Durme. 2017. Discriminative Information Retrieval for Question Answering Sentence Selection. In *15th Conference of the European Chapter of the Association for Computational Linguistics*. 719–725.
[7] Jianpeng Cheng and Mirella Lapata. 2016. Neural Summarization by Extracting Sentences and Words. In *54th Annual Meeting of the Association for Computational Linguistics*. 484–494.
[8] Eunsol Choi, Daniel Hewlett, Jakob Uszkoreit, Illia Polosukhin, Alexandre Lacoste, and Jonathan Berant. 2017. Coarse-to-fine Question Answering for Long Documents. In *55th Annual Meeting of the Association for Computational Linguistics*. 209–220.
[9] François Chollet. 2017. Xception: Deep Learning with Depthwise Separable Convolutions. In *Conference on Computer Vision and Pattern Recognition*.
[10] Yiming Cui, Zhipeng Chen, Si Wei, Shijin Wang, Ting Liu, and Guoping Hu. 2017. Attention-over-attention Neural Networks for Reading Comprehension. In *55th Annual Meeting of the Association for Computational Linguistics*. 593–602.
[11] Mostafa Dehghani, Stephan Gouws, Oriol Vinyals, Jakob Uszkoreit, and Łukasz Kaiser. 2018. Universal Transformers. *arXiv preprint arXiv:1807.03819* (2018).
[12] Mostafa Dehghani, Sascha Rothe, Enrique Alfonseca, and Pascal Fleury. 2017. Learning to attend, copy, and generate for session-based query suggestion. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*. 1747–1756.
[13] Bhuwan Dhingra, Hanxiao Liu, Zhilin Yang, William W Cohen, and Ruslan Salakhutdinov. 2017. Gated-attention Readers for Text Comprehension. In *55th Annual Meeting of the Association for Computational Linguistics*. 1832–1846.
[14] Bhuwan Dhingra, Kathryn Mazaitis, and William W Cohen. 2017. Quasar: Datasets for Question Answering by Search and Reading. *arXiv preprint arXiv:1707.03904* (2017).
[15] Matthew Dunn, Levent Sagun, Mike Higgins, V Ugur Guney, Volkan Cirik, and Kyunghyun Cho. 2017. SearchQA: A New Q&A Dataset Augmented with Context from a Search Engine. *arXiv preprint arXiv:1704.05179* (2017).
[16] Alex Graves. 2013. Generating sequences with recurrent neural networks. *arXiv preprint arXiv:1308.0850* (2013).
[17] Bert F Green Jr, Alice K Wolf, Carol Chomsky, and Kenneth Laughery. 1961. Baseball: an Automatic Question-answerer. In *Western Joint IRE-AIEE-ACM computer conference*. 219–224.
[18] Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching Machines to Read and Comprehend. In *Advances in Neural Information Processing Systems*. 1693–1701.
[19] Phu Mon Htut, Samuel R Bowman, and Kyunghyun Cho. 2018. Training a Ranking Function for Open-Domain Question Answering. In *Conference of the North American Chapter of the Association for Computational Linguistics: Student Research Workshop*. 120–127.
[20] Lukasz Kaiser, Aidan N Gomez, and Francois Chollet. 2018. Depthwise Separable Convolutions for Neural Machine Translation. In *International Conference on Learning Representations*.
[21] Tom Kenter and Maarten de Rijke. 2017. Attentive memory networks: Efficient machine reading for conversational search. In *1st International Workshop on Conversational Approaches to Information Retrieval (CAIR'17)*. ACM.
[22] Tom Kenter, Llion Jones, and Daniel Hewlett. 2018. Byte-level Machine Reading across Morphologically Varied Languages. In *Proceedings of the The Thirty-Second AAAI Conference on Artificial Intelligence (AAAI-18)*.
[23] Ankit Kumar, Ozan Irsoy, Peter Ondruska, Mohit Iyyer, James Bradbury, Ishaan Gulrajani, Victor Zhong, Romain Paulus, and Richard Socher. 2016. Ask Me Anything: Dynamic Memory Networks for Natural Language Processing. In *International Conference on Machine Learning*. 1378–1387.
[24] Souvik Kundu and Hwee Tou Ng. 2018. A Question-Focused Multi-Factor Attention Network for Question Answering. *arXiv preprint arXiv:1801.08290* (2018).
[25] Kenton Lee, Shimi Salant, Tom Kwiatkowski, Ankur Parikh, Dipanjan Das, and Jonathan Berant. 2016. Learning Recurrent Span Representations for Extractive Question Answering. In *International Conference on Learning Representations*.
[26] Yankai Lin, Haozhe Ji, Zhiyuan Liu, and Maosong Sun. 2018. Denoising Distantly Supervised Open-Domain Question Answering. In *56th Annual Meeting of the Association for Computational Linguistics*. 1736–1745.
[27] Kezban Dilek Onal, Ye Zhang, Ismail Sengor Altingovde, Md Mustafizur Rahman, Pinar Karagoz, Alex Braylan, Brandon Dang, Heng-Lu Chang, Henna Kim, Quinten McNamara, Aaron Angert, Edward Banner, Vivek Khetan, Tyler McDonnell, An Thanh Nguyen, Dan Xu, Byron C. Wallace, Maarten de Rijke, and Matthew Lease. 2018. Neural information retrieval: At the end of the early years. *Information Retrieval Journal* 21, 2–3 (June 2018), 111–182.
[28] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100,000+ Questions for Machine Comprehension of Text. In *Conference on Empirical Methods in Natural Language Processing*. 2383–2392.
[29] Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. 2017. Bidirectional Attention Flow for Machine Comprehension. In *International Conference on Learning Representations*.
[30] Yelong Shen, Po-Sen Huang, Jianfeng Gao, and Weizhu Chen. 2017. Reasonet: Learning to Stop Reading in Machine Comprehension. In *23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 1047–1055.
[31] Nitish Srivastava, Geoffrey E 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, 1 (2014), 1929–1958.
[32] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is All You Need. In *Advances in Neural Information Processing Systems*. 5998–6008.
[33] Ellen M Voorhees and others. 1999. The TREC-8 Question Answering Track Report.. In *Text Retrieval Conference*, Vol. 99. 77–82.
[34] Shuohang Wang, Mo Yu, Xiaoxiao Guo, Zhiguo Wang, Tim Klinger, Wei Zhang, Shiyu Chang, Gerald Tesauro, Bowen Zhou, and Jing Jiang. 2018. R3: Reinforced Reader-Ranker for Open-Domain Question Answering. In *The Thirty-Second AAAI Conference on Artificial Intelligence*. 5981–5988.
[35] Shuohang Wang, Mo Yu, Jing Jiang, Wei Zhang, Xiaoxiao Guo, Shiyu Chang, Zhiguo Wang, Tim Klinger, Gerald Tesauro, and Murray Campbell. 2018. Evidence Aggregation for Answer Re-Ranking in Open-Domain Question Answering. In *International Conference on Learning Representations*.
[36] Wenhui Wang, Nan Yang, Furu Wei, Baobao Chang, and Ming Zhou. 2017. Gated Self-Matching Networks for Reading Comprehension and Question Answering. In *55th Annual Meeting of the Association for Computational Linguistics*. 189–198.
[37] Ronald J Williams and David Zipser. 1989. A learning algorithm for continually running fully recurrent neural networks. *Neural computation* 1, 2 (1989), 270–280.
[38] Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, and others. 2016. Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation. *arXiv preprint arXiv:1609.08144* (2016).
[39] Caiming Xiong, Victor Zhong, and Richard Socher. 2017. Dynamic Coattention Networks for Question Answering. In *International Conference on Learning Representations*.
[40] Yichong Xu, Jingjing Liu, Jianfeng Gao, Yelong Shen, and Xiaodong Liu. 2017. Towards Human-level Machine Reading Comprehension: Reasoning and Inference with Multiple Strategies. *arXiv preprint arXiv:1711.04964* (2017).
[41] Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. 2016. Hierarchical Attention Networks for Document Classification. In *Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. 1480–1489.
[42] Adams Wei Yu, David Dohan, Minh-Thang Luong, Rui Zhao, Kai Chen, Mohammad Norouzi, and Quoc V Le. 2018. QANet: Combining Local Convolution with Global Self-Attention for Reading Comprehension. In *International Conference on Learning Representations*.
[43] Yang Yu, Wei Zhang, Kazi Hasan, Mo Yu, Bing Xiang, and Bowen Zhou. 2016. End-to-end Answer Chunk Extraction and Ranking for Reading Comprehension. In *International Conference on Learning Representations*.