Neural Processing Letters
https://doi.org/10.1007/s11063-021-10730-4

Effects of Similarity Score Functions in Attention Mechanisms on the Performance of Neural Question Answering Systems

Yuanyuan Shen[1] · Edmund M.-K. Lai[1] · Mahsa Mohaghegh[1]

Accepted: 20 December 2021

Abstract
Attention mechanisms have been incorporated into many neural network-based natural language processing (NLP) models. They enhance the ability of these models to learn and reason with long input texts. A critical part of such mechanisms is the computation of attention similarity scores between two elements of the texts using a similarity score function. Given that these models have different architectures, it is difficult to comparatively evaluate the effectiveness of different similarity score functions. In this paper, we proposed a baseline model that captures the common components of recurrent neural network-based Question Answering (QA) systems found in the literature. By isolating the attention function, this baseline model allows us to study the effects of different similarity score functions on the performance of such systems. Experimental results show that a trilinear function produced the best results among the commonly used functions. Based on these insights, a new T-trilinear similarity function is proposed which achieved the higher predictive EM and F1 scores than these existing functions. A heatmap visualization of the attention score matrix explains why this T-trilinear function is effective.

Keywords Attention mechanism · Question answering · Deep learning · Natural language processing

1 Introduction

In natural language processing (NLP), handling long sequences of words is a crucial and challenging task. The traditional way is to compress long sequences into short fix-length vectors [1, 2]. However, this can potentially cause a loss of information, especially in the earlier stages of the processing pipeline [3]. More recently, attention mechanisms are used to circumvent this problem by focusing on the information that is most relevant to the target. Their success has made them an indispensable part of neural network-based NLP models for

✉ Edmund M.-K. Lai
edmund.lai@aut.ac.nz

1 School of Engineering, Computer and Mathematical Sciences, Auckland University of Technology, Auckland, New Zealand

machine translation [4, 5], machine reading comprehension [6–9], sentiment analysis [10, 11], and question answering (QA) [12, 13].

The goal of QA systems is to generate answers to questions based on the information provided by a passage. A natural way to do this is to compute the relevance of various parts of the passage to the words in the question. Alternatively, we can say that the words in the question pay attention to different parts of the passage that are considered most relevant. This method has been used successfully in many Recurrent Neural Network (RNN)-based QA systems [6, 7, 14–16].

More recently, another type of attention known as self-attention or intra-attention has been introduced by the Transformer model [17]. The main aim of the Transformer model is to replace recurrent networks by multi-layer feedforward networks. Self-attention is computed in each layer for all inputs to that layer. Its goal is to learn the contextualized word embeddings of a sequence. A position encoder is employed to preserve the knowledge of the order of the words in the input sequence. This architecture has been very successful and inspired many variants, such as OpenAI's GPT [18], GPT-2 [19], GPT-3 [20], and Google's BERT [21]. Another variant, known as Transformer-XL, re-introduced recurrent structures into the model in order to learn longer-term dependencies [22].

Since attention is a core part of these deep learning systems, it is important to understand how the similarity score function affects the performance of these models. However, it is not easy to compare them fairly as the models they are associated with are different. In this paper, we designed a baseline model for this purpose. This model is based on the common characteristics of neural QA systems from the literature that allow us to easily substitute different attention similarity score functions into the model. In this way, we are able to fairly compare the effects of various similarity score functions, both additive [6, 8, 9] and multiplicative [17, 23–27], on the performance of the system.

Using the insights obtained from this comparison, a new function that combines the strengths of the additive and multiplicative methods for similarity calculation is proposed. Results show that this new function provides the highest predictive scores compared with all the existing attention functions. Furthermore, visualization of the attention similarity matrices shows that the proposed method provides a more intuitive understanding of the relationship between the question and the relevant words in the passage.

The rest of this paper is organized as follows. A brief review of neural QA systems that make use of attention mechanisms is provided in Sect. 2. Based on the common characteristics of these systems, in Sect. 3, a baseline QA model is designed for a fair comparison of the different similarity score functions. Section 4 describes the various attention similarity functions considered in this paper. The experimental setup and the comparison results are presented in Sect. 5. In Sect. 6, a new similarity score function is proposed, and its performance is compared with four of the best existing methods. Finally, Sect. 7 presents the conclusions and provides suggestions for future work.

2 Architectures of Neural QA Systems with Attention Mechanisms

Several neural QA systems with attention mechanisms can be found in the literature. Some of them make use of RNNs to capture the contextual dependencies in the word sequences [6, 7], while others encode sequences by fine-tuning the pre-trained language model [17–19, 21, 28]. These neural QA systems can be categorized into two groups—RNN-based and Transformer-based (RNN-free). The attention mechanisms in these two groups of QA models are structured very differently. We shall refer to them as modular attention and self-attention respectively.

A representative of the RNN-based models is BiDAF [6]. It consists of five basic layers. The first layer maps each word to a high-dimensional vector by combining both word-level and character-level embeddings. The second layer is the contextual embedding layer which makes use of a bidirectional Long Short-Term Memory (biLSTM) to capture the temporal dependencies among words for both the passage and the question. This is followed by the attention flow layer that acts as an attention mechanism to generate both context-to-query and query-to-context attention-based summaries. The subsequent modeling layer applies another biLSTM to the output of the attention mechanism, to produce query-aware context representations that capture the temporal interactions among them. The output layer, through two classifiers, provides the prediction of the start and end indices of the answer in the passage.

These layers in the BiDAF model are typical for a range of such QA systems. In some models, several layers are grouped together into a module. An example is the DCN model [7] which has three modules—a document and question encoder, a co-attention encoder, and a dynamic pointing decoder. The first encoder serves the same functions as the first and second layers of BiDAF but without the character-level embeddings. The co-attention encoder acts as an attention mechanism. But it uses a different attention similarity function and a different approach to generate passage-aware query summaries. The decoder performs the same functions as the modelling and output layers of BiDAF. Other examples of such QA systems include FastQA [23], SAN [27], QANet [14], DocQA [29], DrQA [30] and FusionNet [31]. Some of these models include additional linguistic features in addition to word embeddings [23, 27, 30, 31]. DocQA adds an additional self-attention calculation in its attention mechanism. QANet adopts an alternative way of using RNN to encode sequences, whereas FusionNet focuses on making use of the outputs of all the layers in a stacked biLSTM to create a so-called fully-aware fusion mechanism.

Even though the models mentioned above differ from each other in certain ways, there are four aspects that cover all the essential components. First, words are mapped into a high-dimensional vector space called word embeddings. Second, contextual dependencies among the words are captured by an RNN-based context encoder. Third, relevant information from the passage is generated with an attention mechanism. Finally, the output of the attention mechanism is passed into two separate classifiers, which predict the start and end indices of the answer span in the passage respectively. The structure of these models in terms of the four components is shown in Fig. 1.

Transformer-based models are quite different from the RNN-based QA models described above. These models include GPT [18], BERT [21], GPT-2 [19], GPT-3 [20]. The Transformer model that was designed to dispense with the recurrent operations and was originally experimented on machine translation tasks [17]. Hence, this type of models has a feedforward structure. Among them, GPT-3 is the latest and most powerful one. With 175 billion parameters, 10 times more than any previously trained language models, it performs language generation and other capabilities amazingly comparable to humans. Succeeding GPT-3 some large-scaled models have emerged, such as Switch Transformer scaling up to trillion parameters without increasing computational costs [32], DALL-E a Text2Image system trained on text-image pairs [33], and Wu Dao 2.0 have both Chinese and English language generation skills [34].

The architecture of these models typically consists of several encoder or decoder layers. Within each layer is a sub-layer for self-attention, also known as intra-attention. Self-attention allows each position (word) in a sequence to relate to all the other positions, in order to

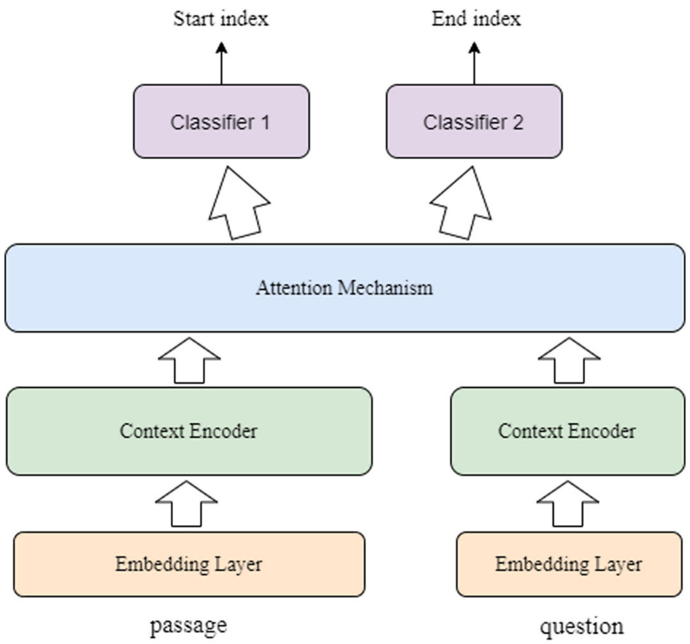


Fig. 1 The relational illustration of the four components

generate a representation of each word in relation to other words in the sequence. Several such layers are stacked to produce a higher-level contextualized representation of the sequence before passing to the downstream related prediction layer. More significantly, as the focus of this group of models is language modeling, there is no such concept of the interaction between the passage and the question. Therefore, the passage and the question are not separated in the earlier layers like that in Fig. 1, but are concatenated as a single sequence of words.

In this paper, we shall focus on modular attention; self-attention will be left as the subject of future work. In particular, we want to investigate the effects of using different similarity score functions. In order to make fair comparisons, we designed a baseline model that captures the invariant aspects of the aforementioned RNN-based neural QA systems. The purpose of this baseline model is not to compete with the state-of-the-art models, but to fulfil the purpose in this work.

3 The Baseline Model

The baseline model is designed to provide a fair comparison of the effectiveness of various attention similarity functions used in modular attention mechanisms. A detailed block diagram of this model is illustrated in Fig. 2. It consists of four modules that are used in a range of QA systems that have been described in Sect. 2. These four modules—input, context encoding, attention, and answer predictor, are described in detail below.

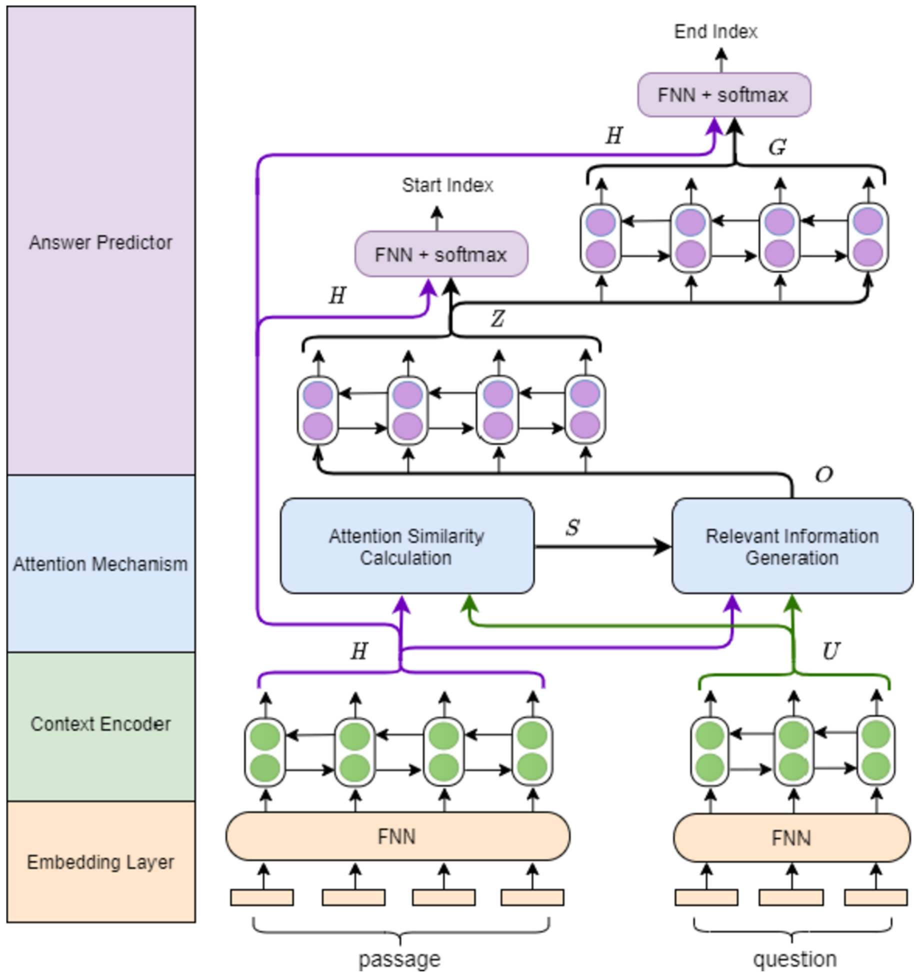


Fig. 2 Block diagram of the baseline model

3.1 Embedding Layer

The task of this layer is to represent each word using its word embedding which is a high-dimensional vector in the Euclidean space. This is a required step in any neural network-based QA model [6, 12, 13, 27, 31, 35, 36]. Given a passage $\{x_1, x_2, \ldots, x_M\}$ with M words, and a question $\{q_1, q_2, \ldots, q_N\}$ with N words. In the word embedding space, the passage is represented as $\left\{e_1^p, e_2^p, \ldots, e_M^p\right\} \in \mathbb{R}^{M\times \upsilon}$ and the question as $\left\{e_1^q, e_2^q, \ldots, e_N^q\right\} \in \mathbb{R}^{N\times \upsilon}$, where υ is the dimension of the word embedding.

Pretrained word embeddings or contextualized word embeddings produced by a pretrained language model can be used, in line with a number of QA models [8, 14, 27, 29]. The feedforward neural network (FNN) is primarily used to reduce the dimension of word embeddings if necessary. If sufficient memory is available, the dimension of the output layer of the FNN can be the same as that of its input layer. Otherwise, its dimension can be made smaller than its input.

3.2 Context Encoder

The sequence of words in the passage is passed to a bi-directional LSTM (biLSTM) layer to fuse the temporal and contextual information into its hidden states to produce the contextual embeddings $\boldsymbol{H} = \{\boldsymbol{h}_t\}_{t=1}^{M} \in \mathbb{R}^{M\times d}$. The sequence of words in the question is fed into the same biLSTM to generate its contextual embeddings $\boldsymbol{U} = \{\boldsymbol{u}_t\}_{t=1}^{N} \in \mathbb{R}^{N\times d}$. The calculation of $\boldsymbol{H}$ and $\boldsymbol{U}$ can be expressed as:

$$\boldsymbol{h}_t = biLSTM\left(\boldsymbol{h}_{t-1}, \boldsymbol{w}_t^{p}\right), \tag{1}$$

$$\boldsymbol{u}_t = biLSTM\left(\boldsymbol{u}_{t-1}, \boldsymbol{w}_t^{q}\right). \tag{2}$$

The advantage of contextual embeddings is that they capture the positional and ordering information of words within a sequence (a sentence or a paragraph) [6, 7, 13, 15]. In other words, contextual embeddings contain the information on the context (what goes before and after) of the current word. As a result, the same word appearing in different places and representing different meaning will have different contextual embeddings.

3.3 Attention Mechanism

As the passage usually contains information that is irrelevant to the question, the main objective of this module is to identify the relevant portions of the passage. An attention mechanism usually consists of two parts: similarity score calculation, and relevant information generation.

3.3.1 Similarity Score Calculation

Given two vectors $\boldsymbol{h}_i$ and $\boldsymbol{u}_j$ representing the ith and jth elements of the contextual embeddings of the passage $\boldsymbol{H}$ and of the question $\boldsymbol{U}$ respectively, the attention similarity score between them is given by

$$s_{ij} = \varphi\left(\boldsymbol{h}_i, \boldsymbol{u}_j\right), \tag{3}$$

where $\varphi(\cdot)$ is the similarity function. The attention score matrix $S \in \mathbb{R}^{M\times N}$ is obtained by arranging these scores in rows and columns.

In this baseline model, we purposedly isolate this computational block from the rest of the model so that φ could be easily replaced in our comparative study. Details of the most common functions are discussed in Sect. 4.

3.3.2 Relevant Information Generation

Instead of using the attention score matrix S directly, it is first normalized. The normalized form of S is known as the attention weight matrix $A \in \mathbb{R}^{M\times N}$ with elements given by

$$\alpha_{ij} = \frac{\exp\left(s_{ij}\right)}{\sum_{k=1}^{N} \exp(s_{ik})}. \tag{4}$$

Such normalization yields $0 \leq \alpha_{ij} \leq 1$, and they represent the probability of each passage word i being related to each question word j. Note that

$$\sum_{j} \alpha_{ij} = 1, \forall i. \tag{5}$$

The context-aware summary $\tilde{\boldsymbol{U}} \in \mathbb{R}^{M \times d}$, also known as the context-to-query summary in some models [6, 29], plays an important role in generating the relevant information in attention mechanisms [6, 23, 37, 38]. It is given by

$$\tilde{\boldsymbol{U}} = A\boldsymbol{U}. \tag{6}$$

The relevant information needed to predict the answer is the function of $\tilde{\boldsymbol{U}}$ and $\boldsymbol{H}$ and it is denoted by $\boldsymbol{O} = \{\boldsymbol{o}_i\}_{i=1}^{M} \in \mathbb{R}^{M \times d_{\mathcal{O}}}$, where

$$\boldsymbol{o}_i = f\big(\boldsymbol{h}_i, \tilde{\boldsymbol{u}}_i\big). \tag{7}$$

f is generally a nonlinear function with trainable weights such as an FNN and $d_{\mathcal{O}}$ is the dimension of the output of f. However, simple concatenation of the operands and their derivatives have been shown to yield good results and are used by a number of QA models [6, 7]. There are two possible concatenation methods:

$$\boldsymbol{o}_i = \big[\boldsymbol{h}_i; \tilde{\boldsymbol{u}}_i\big] \tag{8}$$

with dimension $d_{\mathcal{O}} = 2d$, and

$$\boldsymbol{o}_i = \big[\boldsymbol{h}_i; \tilde{\boldsymbol{u}}_i; \boldsymbol{h}_i \circ \tilde{\boldsymbol{u}}_i\big], \tag{9}$$

where $d_{\mathcal{O}} = 3d$, and $\circ$ denotes the Hadamard product.

3.4 Answer Predictor

The relevant information generated is first passed into a biLSTM to relate the representation at each time step with those preceding and succeeding it. Thus,

$$z_i = biLSTM(\boldsymbol{o}_i, z_{i-1}). \tag{10}$$

Each cell in this biLSTM also plays the important role of projecting the high-dimensional representation from the attention mechanism to a lower-dimensional space. This layer exists in many QA models with attention [6, 7, 27, 29, 38]. It is also known as the modelling layer [6].

The answer is specified by the start and end indices, denoted by p_1 and p_2 respectively, of words from the original passage. p_1 is obtained from the output of a fully connected FNN followed by the softmax function. That is,

$$p_1 = softmax\big(\boldsymbol{w}_{p_1}[\boldsymbol{\mathcal{Z}}; \boldsymbol{H}] + b_{p_1}\big), \tag{11}$$

where $\boldsymbol{\mathcal{Z}} = \{\mathcal{Z}_i\}_{i=1}^{M} \in \mathbb{R}^{M \times 2d}$, $\boldsymbol{w}_{p_1} \in \mathbb{R}^{4d}$ are the weights of the FNN and $b_{p_1} \in \mathbb{R}$ is its bias.

$\boldsymbol{\mathcal{Z}}$ is then fed into another biLSTM to generate $\boldsymbol{G}$. $\boldsymbol{G}$ and $\boldsymbol{H}$ are then passed into another fully connected FNN followed by the softmax function to predict the probability of the end index p_2. This can be expressed in the following equation

$$p_2 = softmax\big(\boldsymbol{w}_{p_2}[\boldsymbol{G}; \boldsymbol{H}] + b_{p_2}\big). \tag{12}$$

As the input to this FNN is $\boldsymbol{G} = \{\boldsymbol{g}_i\}_{i=1}^{M}$ which is derived from $\mathcal{Z}$ via another biLSTM where

$$\boldsymbol{g}_i = biLSTM(z_i, \boldsymbol{g}_{i-1}), \tag{13}$$

the prediction of the end index is conditioned on that of the start index.

4 Similarity Score Functions

As discussed in Sect. 3.3.1, an attention similarity score is computed for the contextual embeddings of the passage and the question. The two groups of similarity score functions, additive and multiplicative, are listed in Table 1.

4.1 Additive Attention Functions

The bilinear function is not seen as an attention similarity function, but according to our preliminary experimental results it provides more insight. Comparing to the trilinear function, bilinear gives much worse results. It is interesting to see how important the element-wise product term is.

The trilinear is well-used in QA models [6, 14, 29], but rarely seen as similarity functions for other uses. It contains three trainable weight vectors $\boldsymbol{w}_p$, $\boldsymbol{w}_q$ and $\boldsymbol{w}_{pq}$, associated with $\boldsymbol{h}_i$, $\boldsymbol{u}_j$ and their element-wise product $\boldsymbol{h}_i \circ \boldsymbol{u}_j$ respectively. Another form of this function

Table 1 Similarity score functions compared in this paper

Group	Types	$\varphi(\boldsymbol{h}_i, \boldsymbol{u}_j)$ forms
Additive	Bilinear*	$\boldsymbol{w}_p\boldsymbol{h}_i + \boldsymbol{w}_q\boldsymbol{u}_j$
	Trilinear [29]	$\boldsymbol{w}_p\boldsymbol{h}_i + \boldsymbol{w}_q\boldsymbol{u}_j + \boldsymbol{w}_{pq}(\boldsymbol{h}_i \circ \boldsymbol{u}_j)$ or $\boldsymbol{w}_\varphi[\boldsymbol{h}_i; \boldsymbol{u}_j; \boldsymbol{h}_i \circ \boldsymbol{u}_j]$
	FNN [8]	$\boldsymbol{w}_\varphi \tanh(\boldsymbol{W}_p\boldsymbol{h}_i + \boldsymbol{W}_q\boldsymbol{u}_j)$
	Concat-FNN [39]	$\boldsymbol{w}_\varphi \tanh(\boldsymbol{W}_p[\boldsymbol{h}_i; \boldsymbol{u}_j])$
Multiplicative	Dot-product [4, 7, 36, 37]	$\boldsymbol{h}_i^T\boldsymbol{u}_j$
	Cosine [26]	$\frac{\boldsymbol{h}_i^T\boldsymbol{u}_j}{\boldsymbol{h}_{i2}\boldsymbol{u}_{j2}}$
	Scaled dot-product [17, 24]	$\frac{\boldsymbol{h}_i^T\boldsymbol{u}_j}{\sqrt{d}}$
	General-1 [4]	$\boldsymbol{h}_i^T\boldsymbol{W}\boldsymbol{u}_j$
	General-2 [25]	$(\boldsymbol{W}_1\boldsymbol{h}_i)^T(\boldsymbol{W}_2\boldsymbol{u}_j)$
	General-3 [27]	$f(\boldsymbol{W}_1\boldsymbol{h}_i), f(\boldsymbol{W}_2\boldsymbol{u}_j)$
	Hadamard [23]	$\boldsymbol{w}^T(\boldsymbol{h}_i \circ \boldsymbol{u}_j)$

Bold indicates conventional expressions for vectors and matrices

$\boldsymbol{w}_p$, $\boldsymbol{w}_q$, $\boldsymbol{w}_{pq}$, $\boldsymbol{w}_\varphi$ and $\boldsymbol{w}$ are trainable vectors. $\boldsymbol{W}_p$, $\boldsymbol{W}_q$, $\boldsymbol{W}$, $\boldsymbol{W}_1$ and $\boldsymbol{W}_2$ are trainable matrices. $f(\cdot)$ is the ReLU activation

*It is not seen as an attention similarity function, but including it helps to gain more insights on the other methods is $\boldsymbol{w}_\varphi\left[\boldsymbol{h}_i; \boldsymbol{u}_j; \boldsymbol{h}_i \circ \boldsymbol{u}_j\right]$, containing only one trainable weight vector that associates with the concatenation of the three vectors. The two forms are the same theoretically, as $\boldsymbol{w}_\varphi$ is just the concatenation of the $\boldsymbol{w}_p$, $\boldsymbol{w}_q$ and $\boldsymbol{w}_{pq}$.

The FNN method first multiplies $\boldsymbol{h}_i$ and $\boldsymbol{u}_j$ with two trainable matrices respectively. From another perspective, this method passes $\boldsymbol{h}_i$ and $\boldsymbol{u}_j$ to two different linear FNN layers. The outputs of these two layers are summed before squashing with the *tanh* function. The resulting vector is then passed into another linear layer to produce the similarity score [8].

The concat-FNN function first concatenates the two input vectors $\boldsymbol{h}_i$ and $\boldsymbol{u}_j$. The concatenation is then fed to a nonlinear FNN to produce the similarity score [3]. Comparing with the FNN function, the concat-FNN requires more computational memory.

4.2 Multiplicative Attention Functions

Dot-product is the most commonly used similarity function for QA models. The dot product between $\boldsymbol{h}_i$ and $\boldsymbol{u}_j$ can be expressed as $\boldsymbol{h}_i^T \boldsymbol{u}_j = |\boldsymbol{h}_i|\left|\boldsymbol{u}_j\right| cos\theta$, where θ is the angle between the two vectors. Hence it is related to the cosine function by the scaling factor $\frac{1}{\|\boldsymbol{h}_i\|_2 \|\boldsymbol{u}_j\|_2}$. The advantage of the cosine function is that its magnitude is limited to within ± 1. It was used early on for neural Turing machines [26] to compute the similarity of two representations. However, none of the QA models have made use of this function.

The Transformer model [17] makes use of the scaled dot-product for its self-attention mechanism. The scaling factor is $\sqrt{d}$, where d is the dimension of the vectors. This scaling apparently promotes efficient learning. As the transformer-based pretrained language models are becoming ubiquitous, the scaled dot-product function is also increasingly utilized [18, 19, 21, 28].

The *general-1* function generalizes the dot product function by introducing a trainable weight matrix $\boldsymbol{W}$. The *general-2* function further incorporates two trainable weight matrices, one for each vector. These two methods make it possible to calculate the attention score for the case where the two vectors have different dimensions. The *general-3* method rectifies the linearly transformed input vectors with the ReLU activation function before computing the dot-product. The $ReLU$ function is defined as $f(x) = \max(0, x)$.

The Hadamard method has never been used as a similarity function of attention. Instead, it was used to calculate the similarity for generating word-in-question features to augment word embedding representations [23]. It has been included as it is a bit different from the other multiplicative similarity functions. Here, $\boldsymbol{w}$ is a trainable weight vector.

5 Experiments for Comparing the Effects of Similarity Functions

With the baseline model, we are able to fairly evaluate the effects of the similarity score functions on the performance of QA systems. Details of the experimental setup and the results are described in this Section.

5.1 Dataset and Evaluation Metrics

Our experiments make use of the SQuAD dataset which is a popular benchmark for testing QA and machine reading comprehension systems [40]. Its passages were collected from a wide range of articles on Wikipedia. The answer to each question can be found in the corresponding passage, identified by the text span in the passage. This dataset contains 107,785 question–answer pairs obtained by the crowdsourcing based on the articles. Each answer to a question is the text span from the corresponding passage. It covers a range of question types.

Two evaluation metrics are used—EM and F1. EM is a metric that measures the rate or percentage of exact match with the ground truth. F1 measures the harmonic mean of precision and recall. For each question, precision is the ratio of the number of correct words and the number of words in the predicted answer. Recall is calculated as the number of correct words divided by the number of words in the ground truth answer. The F1 score is computed per question and then averaged across all questions.

5.2 Training Setup

The baseline models in our experiment are trained using the following setup parameters. The word embeddings are 300-dimensional GloVe vectors[1] pre-trained on 840 billion tokens in Common Crawl [41]. The out-of-vocabulary (OOV) words are set to be zero vectors. The output dimension of the FNN in the input module is set to be 100. The training batch is set to be 60, and the number of epochs is 20. We use the Adadelta optimizer [42] with an initial learning rate of 0.5. A 0.2 dropout rate is applied to the input for each RNN and linear layers, except for the linear layers in the answer prediction module. An exponential moving average (EMA) of all the trainable weights with a decay rate of 0.999 is utilised. In the testing stage, these averages are used for predicting the answers. The sum of the cross-entropy functions, one for the start index prediction and the other for the end index prediction, are used as the loss function.

5.3 Results

Using the eleven similarity functions listed in Table 1, results are obtained using the two different ways of generating the output of the attention mechanism, given by (8) and (9). For the sake of expression convenience, we denote results using (8) as O_1, and those using (9) as O_2. The best EM and F1 scores for models obtained from five training runs are shown in Table 2. Figure 3 shows box and whisker plots of the corresponding EM scores. The red circles and orange horizontal lines indicate the average and median values respectively.

Comparing the results of O_1 and O_2, firstly, the results show that O_2 gives better results for both EM and F1 with all similarity score functions except *general-2*. This indicates the element-wise product between the contextual passage embeddings and the context-aware query representations in O_2 helps to improve the performance. The possible reason is that the element-wise product of two vectors contains some information that neural networks cannot learn from the concatenation of the two vectors. Remember that the output of the attention mechanism is passed into a recurrent neural network.

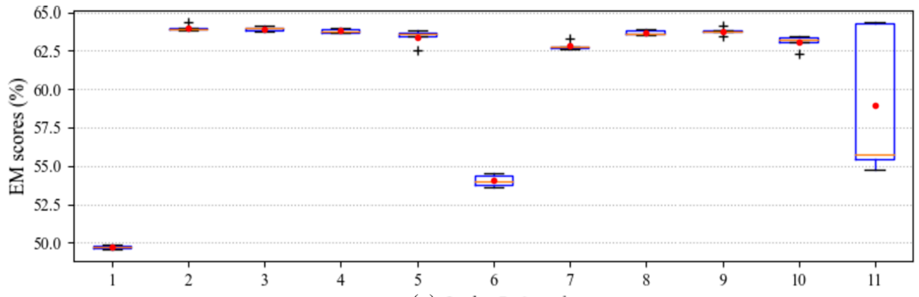
Secondly, the Hadamard function is the best performer overall, with the EM score only slightly below *general-3* for O_2. Figure 3 shows that the performance of the models based on this similarity function for O_1 is not robust across different runs—EM scores ranging from 55.0 to 64.3 and F1 scores from 67.0 to 75.5. However, no such problem exists using O_2. This shows that the inclusion of the $\boldsymbol{h}_i \circ \tilde{\boldsymbol{u}}_i$ term plays a vital role in improving the robustness of the model with this similarity score function.

1 downloaded from https://nlp.stanford.edu/projects/glove/.

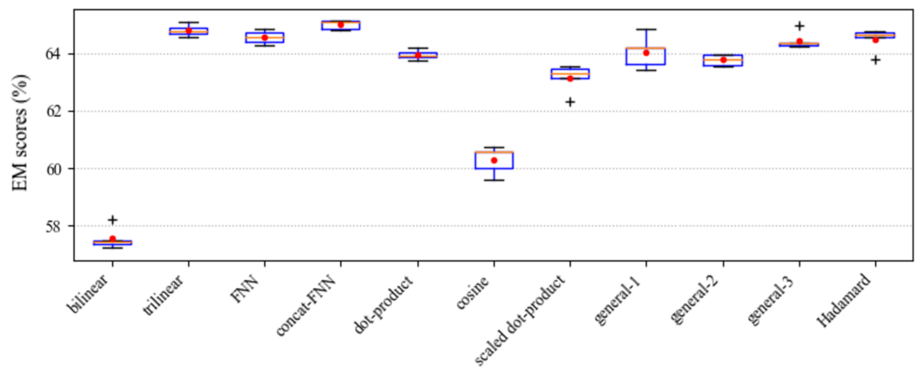
Table 2 EM and F1 scores with different attention similarity score functions for O_1 and O_2

Groups	Similarity score function	O_1		O_2	
		EM	F1	EM	F1
Additive	bilinear	49.9	60.6	58.2	69.0
	trilinear	**64.4**	75.0	**65.1**	**75.5**
	FNN	64.1	74.6	64.8	75.2
	concat-FNN	64.0	74.8	**65.1**	75.4
Multiplicative	dot-product	63.8	74.5	64.2	75.0
	cosine	54.5	65.7	60.7	71.8
	scaled dot-product	63.3	74.0	63.5	74.4
	general-1	63.9	74.5	64.8	75.0
	general-2	64.1	74.9	64.0	74.7
	general-3	63.5	74.2	64.9	75.3
	Hadamard	64.3	**75.2**	64.7	75.3

Bold indicates the maximum value in each column



(a) In the O_1 setting



(b) In the O_2 setting

Fig. 3 The box plots of EM scores under O_1 and Q_2

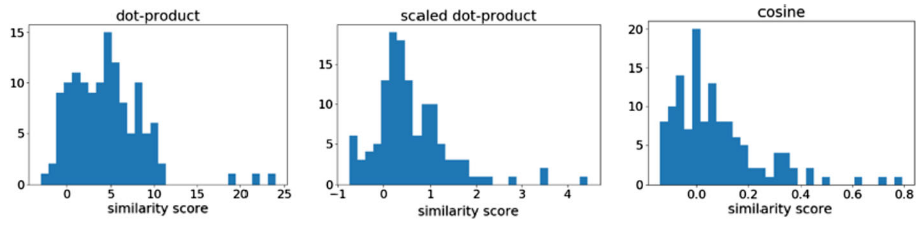


Fig. 4 Histograms of the Similarity Scores for the three functions without parameters

Comparing the results of additive and multiplicative groups, the highest scores are mainly given by the additive group, especially the trilinear attention function. The bilinear has the two terms $\boldsymbol{w}_p\boldsymbol{h}_i$ and $\boldsymbol{w}_q\boldsymbol{u}_j$ which are also contained in trilinear. However, because of the lack of the $\boldsymbol{h}_i \circ \boldsymbol{u}_i$ term, bilinear gives dramatically low scores. This suggests the relation between the two input vectors are not bilinear, and including the $\boldsymbol{h}_i \circ \boldsymbol{u}_i$ term into the bilinear transformation improves the predictive accuracy of the model significantly. Another possible reason is that with more parameters trilinear has more capacity for the higher accuracy, which aligns with the observations in the literature on network design space [43, 44]. Although the FNN function has similar parameters, it is more nonlinear than linear. Its slightly lower results comparing to the trilinear function suggests that the information of the element-wise product of two vectors may not be fully learned through passing the concatenation of the same two vectors into a feedforward neural network.

It is interesting to note that the scaled dot-product function, a default preference in pre-trained language models, performs the same or slightly worse than the simple dot-product. This could be due to the scaling factor, defined as the squared root of the dimensionality of the vectors, which is quite large. This may cause the magnitudes of the similarity score to be reduced to such an extent that their values are too close to each other, especially after being normalised by the softmax function. In order to confirm this hypothesis, we extract the similarity scores of dot-product, scaled dot-product along with another non-parameter function cosine for the same test sample from their corresponding trained models. The histograms of their similarity scores are demonstrated in Fig. 4. It clearly shows that the scaling factor in the scaled dot-product function does reduce the range of the scores significantly compared to dot-product.

Another function whose values are restricted to a small finite range—the cosine function, gives the poorest results. Thus, it is clear that based on this type of system architecture, the similarity function should be able to produce a larger range of values.

6 Proposed New Attention Similarity Function

Based on the findings presented in the previous Section, we seek to obtain a better attention similarity function by combing the strengths of both the additive and multiplicative similarity functions.

In the additive group, the trilinear function achieves the best results among all the similarity functions, while the bilinear shows the worst performance. The only difference between these two functions is the element-wise $\boldsymbol{h}_i \circ \boldsymbol{u}_i$ associated term. Projecting the element-wise product along with its corresponding two input vectors helps to improve the performance of the model significantly.

In the multiplicative group, the *ReLU* function that is used to transform the two input vectors in the general-3 function on top of general-2 helps to improve the performance noticeably in the O_2 setting. *ReLU* is the rectified linear activation function that outputs the input unchangeably if it is positive, otherwise outputs 0. Therefore, we introduce a *ReLU* transformation operation into the best similarity score function, trilinear, to propose a new attention function called T-trilinear (short for transformed trilinear):

$$\boldsymbol{h}_i^{\mathrm{t}} = ReLU(\boldsymbol{W}_1\boldsymbol{h}_i + \boldsymbol{b}_1) \tag{14}$$

$$\boldsymbol{u}_i^{\mathrm{t}} = ReLU(\boldsymbol{W}_2\boldsymbol{u}_i + \boldsymbol{b}_2) \tag{15}$$

$$\varphi(\boldsymbol{h}_i, \boldsymbol{u}_j) = \boldsymbol{w}_p\boldsymbol{h}_i^{\mathrm{t}} + \boldsymbol{w}_q\boldsymbol{u}_i^{\mathrm{t}} + \boldsymbol{w}_{pq}(\boldsymbol{h}_i^{\mathrm{t}} \circ \boldsymbol{u}_i^{\mathrm{t}}) \tag{16}$$

where $\boldsymbol{W}_1$ and $\boldsymbol{W}_2 \in \mathbb{R}^{d\times d}$, $\boldsymbol{b}_1$ and $\boldsymbol{b}_2 \in \mathbb{R}^{d\times 1}$, $\boldsymbol{w}_p$, $\boldsymbol{w}_q$ and $\boldsymbol{w}_{pq} \in \mathbb{R}^{d\times 1}$.

The two input vectors $\boldsymbol{h}_i$ and $\boldsymbol{u}_i$ are passed into an FNN with the *ReLU* activation before going through the trilinear function to get the similarity score. The *ReLU* function deactivates the neurons to get the desired output while keeping the efficiency of the computation.

6.1 Evaluation Results

The EM and F1 scores of the T-trilinear function, along with all the functions from the additive group except for bilinear and *general-3* from the multiplicative group whose results are the top ones shown in Table 2, are demonstrated in Table 3. It can be seen that the proposed T-trilinear method achieves the best results under both O_1 and O_2 settings. Figure 5 shows the distributions of the results through different runs. It demonstrates that the performance of the T-trilinear function is consistent among runs, particularly in comparison with the Hadamard function.

The loss curves, which show the training convergence behaviour, with the similarity score functions listed in Table 3 are shown in Fig. 6. It can be observed that all the functions converge faster in the O_2 setting than in the O_1 setting. In the O_1 setting, the one that uses the trilinear function converges most rapidly in the early stage, followed by general-3 slightly slower. All the other ones demonstrate quite similar convergence patterns. The proposed T-trilinear function shows a similar trend to FNN and concat-FNN in the initial stages. This is not surprising as the three have relatively more trainable weights and the models need to search for the optimal combination. After that, their losses continue to decrease and end up

Table 3 Comparison of EM and F1 scores for the best performing attention similarity functions

Attention similarity function	O_1		O_2	
	EM	F1	EM	F1
trilinear	64.4	75.0	65.1	75.5
FNN	64.1	74.6	64.8	75.2
concat-FNN	64.0	74.8	65.1	75.4
general-3	63.5	74.2	64.9	75.3
T-trilinear	**64.5**	**75.2**	**65.3**	**75.8**

Bold indicates the maximum value in each column

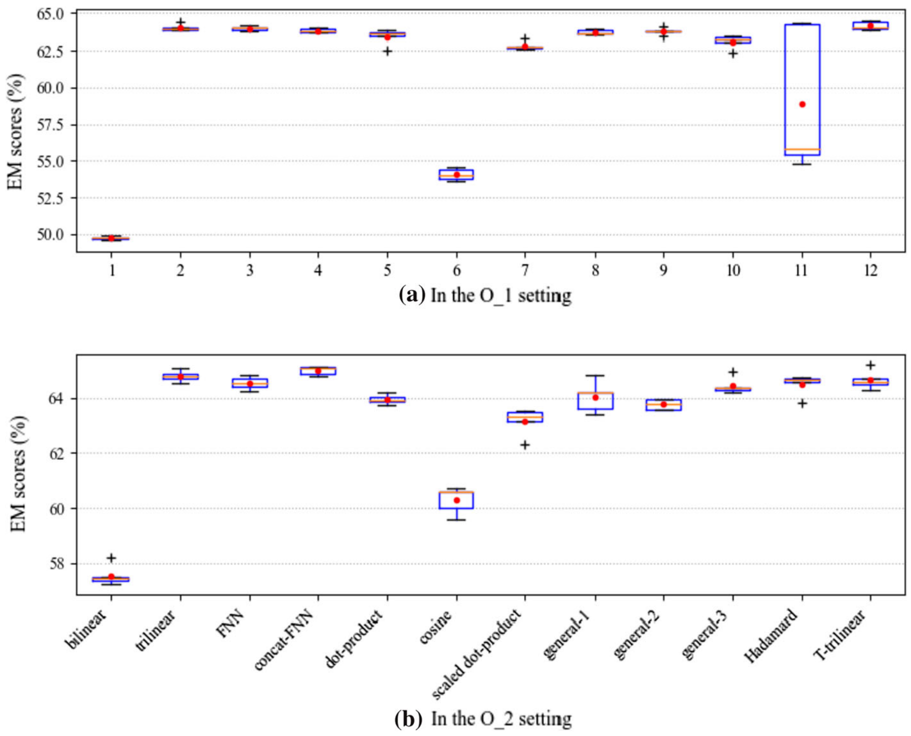


Fig. 5 The box plots of EM and F1 scores under O_1 and Q_2

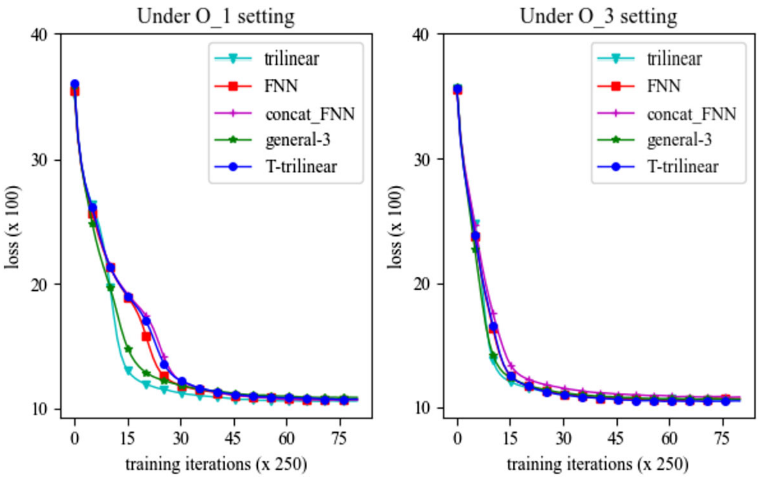


Fig. 6 Loss curves of the baseline model with the similarity functions listed in Table 3

Table 4 EM and F1 scores of three QA models with their original attention similarity score functions in comparison with the proposed T-trilinear function

Models	Original similarity function	Original similarity function		T-trilinear	
		EM	F1	EM	F1
BiDAF	trilinear	67.7	77.3	68.5	77.9
DCN	dot	65.4	75.6	65.7	76.1
QANet	trilinear	73.6	82.7	74.9	83.1

with a similar level as the other two functions. In the O_2 setting, all the functions display similar convergence trends and T-trilinear reaches a relatively lower loss in the later stage.

In order to confirm that the proposed T-trilinear attention function works well with the original QA models, it was implemented on BiDAF, DCN and QANet. The original similarity functions for both BiDAF and QANet is the trilinear function and DCN uses the dot function. The comparison results in Table 4 show that using the T-trilinear function, both the EM and F1 scores are higher for all three models. These results confirm that the T-trilinear function is indeed more effective compared with the other similarity functions.

Taking a closer look at the extents of performance improvement in Tables 3 and 4, we can see that the improvement in the three existing models is more significant than that of the baseline model. This indicates that the proposed T-trilinear function may work better in "real" models. This is likely due to additional components included in the "real" models but not in the baseline model as the latter only consists of the four invariant modules of the existing RNN-based neural QA models.

Comparing the improvement extents among the three existing models, we can see that the performance of the QANet model is improved more than the other two models. The DCN model demonstrates a slightly lower improvement than BiDAF does. This is very likely caused by the different aspects that these models distinct themselves from each other. In the future research, the baseline model will be used to study the effectiveness of different approaches for other modules.

6.2 Visualization and Local Explanation

As the proposed T-trilinear is inspired by the strengths of trilinear and general-3 discussed in Sect. 6, this section visually compares the three using a test sample chosen from the dataset. Figure 7 shows the sample, consisting of the passage, the question, and the ground truth answer. We plot the similarity scores between individual words in the passage and those in the question as heatmaps in Fig. 8. The x-axis of the heatmap are words from the question and the y-axis are words from the passage. The colour of each cell in a heatmap represents the attention similarity score between the corresponding words in the x and y axes. The darker the colour is, the higher the score is. Results in both the O_1 and O_2 settings are shown.

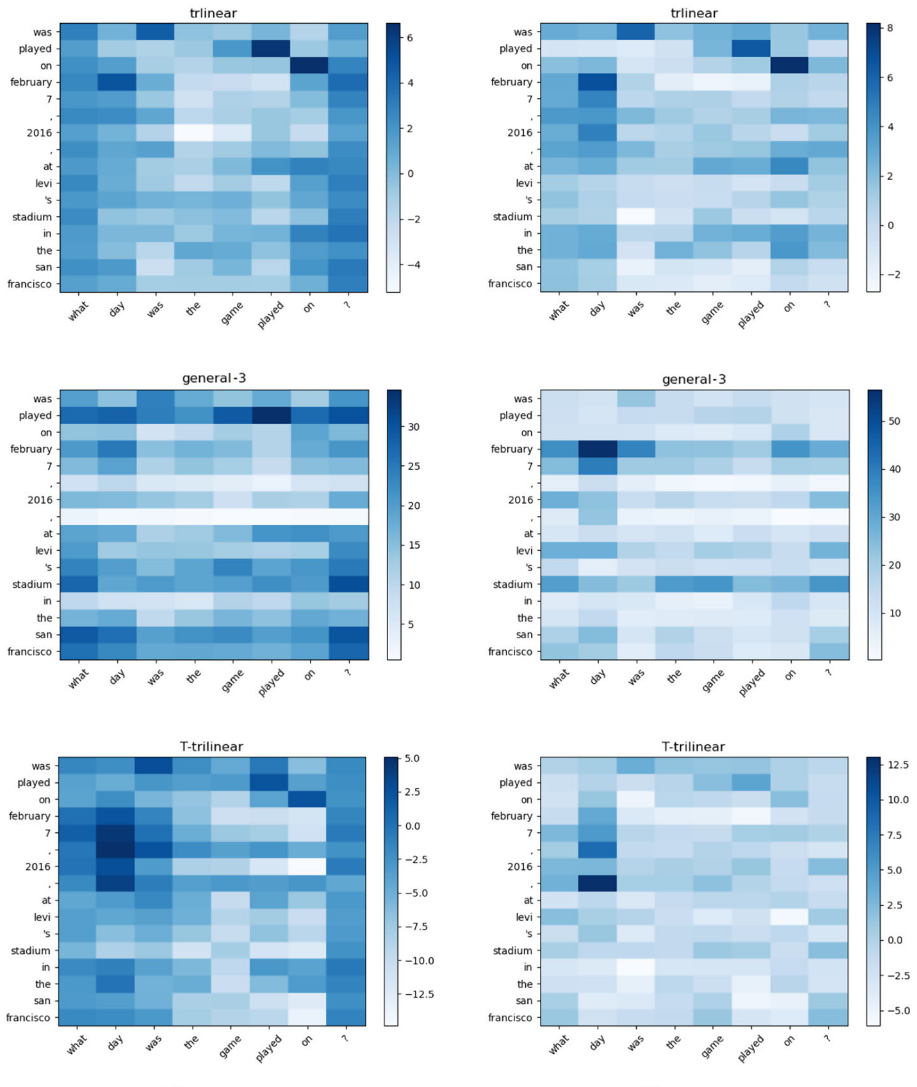
Intuitively the word "day" in the question should be more related to the word span "*February 7, 2016"* in the passage. Consider Fig. 8a which is for the O_1 setting. For the trilinear and general-3 functions, the cells corresponding to the answer words *February 7, 2016* display no distinct darker colour compared with other cells. This shows that these two attention functions fail to pay attention to the related text span in the passage. However,

… The American football conference (AFC) champion Denver broncos defeated the national football conference (NFC) champion Carolina panthers 24–10 to earn their third super bowl title. the game was played on February 7, 2016, at Levi's stadium in the san Francisco bay area at Santa Clara, California. …

Question: What day was the game played on?

True answer: February 7, 2016

Fig. 7 The chosen test sample of passage-question–answer



(a) In O_1 setting

(b) In O_2 setting

Fig. 8 Heatmaps of the attention similarity matrices

the heatmap of our proposed T-trilinear function clearly has darker colour in these cells. This shows that the T-trilinear function is more able to pay attention to the right text span in the passage.

With the O_2 setting in Fig. 8b, trilinear's darker color seems to be able to concentrate on a smaller number of areas than in the O_1 setting. It shows slight darker on the answer word range *February 7, 2016*, but still has a relatively high focus on other words such as *was*, *played* and *on*, which are irrelevant to the question. The general-3 exhibits a very different heatmap from O_1. It shows high scores for the relevant words in the passage, especially *February 7*, but the pattern still seems row-wise (passage-word-wise) like in O_1. The T-trilinear method can focus on the answer words clearly and pay less attention to words that are irrelevant to the answer except for the dot behind the answer span. This may be because part of the answer information is transferred to the adjacent position (the dot behind 2016) of the answer span through the RNNs in the model.

7 Conclusions

Attention mechanisms are prevalent techniques for neural network-based NLP models in recent years. Attention similarity calculation is an important part of attention mechanisms. In this work we compared several similarity calculation functions that have been used in various NLP tasks. We tested them in the QA context, using a QA baseline model we created, benchmarked on the SQuAD dataset.

The experimental results demonstrate that the additive similarity functions perform better than the multiplicative ones. The trilinear similarity function in the additive group achieved the highest predictive scores. The general-3 function, applying the *ReLU* operation on top of general-2, achieves the highest EM and F1 scores in the O_2 setting. As for the Hadamard similarity function, it achieved the highest predictive scores in the O_1 setting compared with those based on the inner product and has competitive scores as general-3 in O_2. However, its results are not consistent across multiple training runs in O_1. The two most commonly used functions—dot-product and scaled dot-product functions are among the worst performers.

Based on these results, we proposed a new function T-trilinear, which introduces the *ReLU* transformation applied in general-3 to trilinear. Experimental results show that T-trilinear demonstrates the highest predictive scores as well as has stable performance across multiple runs.

In this paper, we focused on the extractive QA task, which means that the answer is a consecutive text span in the passage. Investigating the effects of the similarity functions for multiple-step reasoning tasks would be a reasonable next step. It is also worthwhile to test their performance in other NLP areas, such as textual similarity prediction and neural machine translation.

Acknowledgements The authors would like to thank New Zealand's Callaghan Innovation for the R&D Fellowship Grants, and Ambit AI Limited in New Zealand for the sponsorship.

Funding Open Access funding enabled and organized by CAUL and its Member Institutions.

References

1. Cho K, Van Merriënboer B, Gulcehre C, Bahdanau D, Bougares F, Schwenk H, Bengio Y (2014) 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). Doha, Qatar. https://doi.org/10.3115/v1/D14-1179
2. Sutskever I, Vinyals O, Le QV (2014) Sequence to sequence learning with neural networks. In: Proceedings of the 27th international conference on neural information processing systems (NIPS'14). Montreal, Canada
3. Cho K, van Merrienboer B, Bahdanau D, Bengio Y (2014) On the properties of neural machine translation: Encoder-decoder approaches. In: Proceedings of the 8th workshop on syntax, semantics and structure in statistical translation (SSST-8). Doha, Qatar. https://doi.org/10.3115/v1/W14-4012
4. Luong M-T, Pham H, Manning C D (2015) Effective approaches to attention-based neural machine translation. In: Proceedings of the 2015 conference on empirical methods in natural language processing (EMNLP). Lisbon, Portugal. https://doi.org/10.18653/v1/D15-1166
5. Zhang B, Xiong D, Xie J, Su J (2020) Neural machine translation with gru-gated attention model. IEEE Trans Neural Netw Learn Syst 31:4688–4698. https://doi.org/10.1109/TNNLS.2019.2957276
6. Seo M, Kembhavi A, Farhadi A, Hajishirzi H (2017) Bidirectional attention flow for machine comprehension. In: Proceedings of the 5th international conference on learning representations (ICLR). Toulon, France
7. Xiong C, Zhong V, Socher R (2017) Dynamic coattention networks for question answering. In: Proceedings of the 5th international conference on learning representations (ICLR). Toulon, France
8. Wang W, Yang N, Wei F, Chang B, Zhou M (2017) Gated self-matching networks for reading comprehension and question answering. In: Proceedings of the 55th annual meeting of the association for computational linguistics (Volume 1: Long Papers). Vancouver, Canada. https://doi.org/10.18653/v1/P17-1018
9. Wang S, Jiang J (2016) Machine comprehension using match-lstm and answer pointer. In: Proceedings of the 5th international conference on learning representations (ICLR). Toulon, France
10. Wang J, Sun C, Li S, Liu X, Si L, Zhang M, Zhou G (2019) Aspect sentiment classification towards question-answering with reinforced bidirectional attention network. In: Proceedings of the 57th annual meeting of the association for computational linguistics. Florence, Italy. https://doi.org/10.18653/v1/P19-1345
11. Sadr H, Pedram MM, Teshnehlab M (2019) A robust sentiment analysis method based on sequential combination of convolutional and recursive neural networks. Neural Process Lett 50:2745–2761. https://doi.org/10.1007/s11063-019-10049-1
12. Kumar A, Irsoy O, Ondruska P, Iyyer M, Bradbury J, Gulrajani I, Zhong V et al (2016) Ask me anything: dynamic memory networks for natural language processing. In: Proceedings of the 33rd international conference on machine learning. New York, USA
13. Xiong C, Merity S, Socher R (2016) Dynamic memory networks for visual and textual question answering. In: Proceedings of the 33rd international conference on machine learning. New York, USA
14. Yu AW, Dohan D, Luong M-T, Zhao R, Chen K, Norouzi M, Le QV (2018) QANet: combining local convolution with global self-attention for reading comprehension. In: Proceedings of the 6th international conference on learning representations (ICLR). Vancouver, Canada
15. Feldman Y, El-Yaniv R (2019) Multi-hop paragraph retrieval for open-domain question answering. In: Proceedings of the 57th annual meeting of the association for computational linguistics. Florence, Italy. https://doi.org/10.18653/v1/P19-1222
16. Li R, Jiang Z, Wang L, Lu X, Zhao M (2020) Directional attention weaving for text-grounded conversational question answering. Neurocomputing 391:13–24. https://doi.org/10.1016/j.neucom.2020.01.056
17. Vaswani A, Shazeer N, Parmar N, Uszkoreit J, Jones L, Gomez A N, Kaiser Ł et al (2017) Attention is all you need. In: Proceedings of the 31st international conference on neural information processing systems (NIPS'17). Long Beach, California, USA
18. Radford A, Narasimhan K, Salimans T, Sutskever I (2018) Improving language understanding by generative pre-training. OpenAI blog

19. Radford A, Wu J, Child R, Luan D, Amodei D, Sutskever I (2019) Language models are unsupervised multitask learners. OpenAI blog
20. Brown TB, Mann B, Ryder N, Subbiah M, Kaplan J, Dhariwal P, Neelakantan A et al (2020) Language models are few-shot learners. arXiv preprint arXiv:200514165
21. Devlin J, Chang M-W, Lee K, Toutanova K (2019) Bert: pre-training of deep bidirectional transformers for language understanding. In: Proceedings of the 2019 annual conference of the North American chapter of the association for computational linguistics (NAACL-HLT). Minneapolis, Minnesota, USA. https://doi.org/10.18653/v1/N19-1423
22. Dai Z, Yang Z, Yang Y, Carbonell J, Le Q, Salakhutdinov R (2019) Transformer-xl: attentive language models beyond a fixed-length context. In: Proceedings of the 57th annual meeting of the association for computational linguistics. Florence, Italy. https://doi.org/10.18653/v1/P19-1285
23. Weissenborn D, Wiese G, Seiffe L (2017) Making neural QA as simple as possible but not simpler. In: Proceedings of the 21st conference on computational natural language learning (CoNLL 2017). Vancouver, Canada. https://doi.org/10.18653/v1/K17-1028
24. Geng X, Wang L, Wang X, Qin B, Liu T, Tu Z (2020) How does selective mechanism improve self-attention networks?. In: Proceedings of the 58th annual meeting of the association for computational linguistics. https://doi.org/10.18653/v1/2020.acl-main.269
25. Britz D, Goldie A, Luong M-T, Le Q (2017) Massive exploration of neural machine translation architectures. In: Proceedings of the 2017 conference on empirical methods in natural language processing (EMNLP). Copenhagen, Denmark. https://doi.org/10.18653/v1/D17-1151
26. Graves A, Wayne G, Danihelka I (2014) Neural turing machines. arXiv preprint arXiv:14105401
27. Liu X, Shen Y, Duh K, Gao J (2018) Stochastic answer networks for machine reading comprehension. In: Proceedings of the 56th annual meeting of the association for computational linguistics (Volume 1: Long Papers). Melbourne, Australia. https://doi.org/10.18653/v1/P18-1157
28. Brown T B, Mann B, Ryder N, Subbiah M, Kaplan J, Dhariwal P, Neelakantan A et al (2020) Language models are few-shot learners. CoRR abs/2005.14165
29. Clark C, Gardner M (2018) Simple and effective multi-paragraph reading comprehension. In: Proceedings of the 56th annual meeting of the association for computational linguistics (Volume 1: Long Papers). Melbourne, Australia. https://doi.org/10.18653/v1/P18-1078
30. Chen D, Fisch A, Weston J, Bordes A (2017) Reading wikipedia to answer open-domain questions. In: Proceedings of the 55th annual meeting of the association for computational linguistics (Volume 1: Long Papers). Vancouver, Canada. https://doi.org/10.18653/v1/P17-1171
31. Huang H-Y, Zhu C, Shen Y, Chen W (2018) Fusionnet: fusing via fully-aware attention with application to machine comprehension. In: Proceedings of the 6th international conference on learning representations (ICLR). Vancouver, Canada
32. Fedus W, Zoph B, Shazeer N (2021) Switch transformers: scaling to trillion parameter models with simple and efficient sparsity. arXiv preprint arXiv:210103961
33. Ramesh A, Pavlov M, Goh G, Gray S, Voss C, Radford A, Chen M et al (2021) Zero-shot text-to-image generation. arXiv preprint arXiv:210212092
34. Romero A (2021) Gpt-3 scared you? Meet wu dao 2.0: a monster of 1.75 trillion parameters
35. Kobayashi S, Tian R, Okazaki N, Inui K (2016) Dynamic entity representation with max-pooling improves machine reading. In: Proceedings of the 2016 conference of the North American chapter of the association for computational linguistics: human language technologies. San Diego, California, USA. https://doi.org/10.18653/v1/N16-1099
36. Sukhbaatar S, Weston J, Fergus R (2015) End-to-end memory networks. In: Proceedings of the 28th international conference on neural information processing systems (NIPS'15). Montreal, Canada
37. Xiong C, Zhong V, Socher R (2018) DCN+: mixed objective and deep residual coattention for question answering. In: Proceedings of the 6th international conference on learning representations (ICLR). Vancouver, Canada
38. Wang Y, Liu K, Liu J, He W, Lyu Y, Wu H, Li S et al (2018) Multi-passage machine reading comprehension with cross-passage answer verification. In: Proceedings of the 56th annual meeting of the association for computational linguistics (Volume 1: Long Papers). Melbourne, Australia. https://doi.org/10.18653/v1/P18-1178
39. Bahdanau D, Cho K, Bengio Y (2015) Neural machine translation by jointly learning to align and translate. In: Proceedings of the 3rd international conference on learning representations (ICLR). San Diego, CA, USA
40. Rajpurkar P, Zhang J, Lopyrev K, Liang P (2016) Squad: 100,000+ questions for machine comprehension of text. In: Proceedings of the 2016 conference on empirical methods in natural language processing. Austin, Texas, USA. https://doi.org/10.18653/v1/D16-1264

41. Pennington J, Socher R, Manning C (2014) Glove: global vectors for word representation. In: Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP). Doha, Qatar. https://doi.org/10.3115/v1/D14-1162
42. Zeiler MD (2012) Adadelta: an adaptive learning rate method. arXiv preprint arXiv:12125701
43. Radosavovic I, Johnson J, Xie S, Lo W-Y, Dollár P (2019) On network design spaces for visual recognition. In: Proceedings of the IEEE/CVF international conference on computer vision
44. Radosavovic I, Kosaraju RP, Girshick R, He K, Dollár P (2020) Designing network design spaces. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition