MRNN: A Multi-Resolution Neural Network with Duplex Attention for Deep Ad-Hoc Retrieval

Tolgahan Cakaloglu[1,2], Xiaowei Xu[2] and Roshith Raghavan[3]
[1]*Walmart Labs, Dallas, Texas, U.S.A.*
[2]*University of Arkansas, Little Rock, Arkansas, U.S.A.*
[3]*CVS Health, Boston, Massachussets, U.S.A.*

Keywords: Deep Learning, Ad-Hoc Retrieval, Learning Representations, Ranking, Text Matching.

Abstract: The primary goal of ad-hoc retrieval is to find relevant documents satisfying the information need posted by a natural language query. It requires a good understanding of the query and all the documents in a corpus, which is difficult because the meaning of natural language texts depends on the context, syntax, and semantics. Recently deep neural networks have been used to rank search results in response to a query. In this paper, we devise a *multi-resolution neural network* (*MRNN*) to leverage the whole hierarchy of representations for ad-hoc retrieval. The proposed MRNN model is capable of deriving a representation that integrates representations of different levels of abstraction from all the layers of the learned hierarchical representation. Moreover, a duplex attention component is designed to refine the multi-resolution representation so that an optimal context for matching the query and document can be determined. More specifically the first attention mechanism determines optimal context from the learned multi-resolution representation for the query and document. The latter attention mechanism aims to fine-tune the representation so that the query and the relevant document are closer in proximity. The empirical study shows that MRNN with the duplex attention is significantly superior to existing models used for ad-hoc retrieval on benchmark datasets including SQuAD, WikiQA, QUASAR, and TrecQA.

1 INTRODUCTION

Ad-hoc retrieval (Voorhees and Harman, 2005) allows a user to specify the information need using a natural language query, which is instrumental for many applications including question answering and information retrieval. Traditional approach uses simple statistical features such as term frequency and document frequency to represent the query and the document (Salton and McGill, 1986). The query and documents are matched by using some similarity measure like cosine similarity. However, this approach is less effective because the representation doesn't consider the rich context of texts.

Recently deep neural networks have been used to rank search results in response to a query for ad-hoc retrieval (Palangi et al., 2016) (McDonald et al., 2018). The fundamental idea of deep learning is that a hierarchical representation is learned automatically, where each layer is a representation that is a high-level abstraction of the representation from the previous layer. The most abstract representation from the last layer of the hierarchy is then used for the machine learning task. However, an abstract representation from a single layer or only a few layers may not be able to capture the semantic relationship of concepts across different levels of an ontology. A text may contain concepts from different levels of the ontology. For example, *"a banana is a fruit"* where *"fruit"* is a high level concept comparing to *"banana"*.

In this paper, we present a new *Multi-Resolution Neural Network* for ad-hoc retrieval, called *MRNN*, which leverages representations across all levels of abstractions in the learned hierarchical representation. The learned representation achieves a multi-resolution effect that can represent concepts and their relationships across all the levels. As illustrated in Figure 1 the proposed model consists of the following components:

- *Multi-Resolution Feature Maps*, which transforms the input query and document into a multi-resolution representation.
- *Duplex Attention*, which implements two attention mechanisms to refine the multi-resolution repre-

Cakaloglu, T., Xu, X. and Raghavan, R.
MRNN: A Multi-Resolution Neural Network with Duplex Attention for Deep Ad-Hoc Retrieval.
DOI: 10.5220/0011896300003393
In *Proceedings of the 15th International Conference on Agents and Artificial Intelligence (ICAART 2023) - Volume 2*, pages 466-477
ISBN: 978-989-758-623-1; ISSN: 2184-433X

sentation. The first attention mechanism determines an optimal context based on *n*-Grams from the learned multi-resolution representation for the query and document. The latter attention mechanism aims to fine-tune the representation so that the query and the relevant document are closer in proximity.

- *Aggregation*, which calculates the similarity between the pair of query and document through aggregation of the duplex attention as the input to the loss function.
- *Distance Metric Loss*, which is a loss function used to train the model by minimizing the loss.

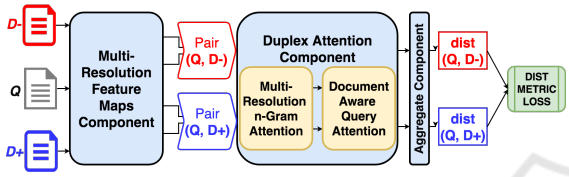

Figure 1: Overall training flow diagram for the proposed Multi-Resolution Neural Network with a query Q, a positive document D^+ and a negative document D^-.

The main contribution of the paper is as follows:

- We first propose a new deep learning model MRNN that leverages representations across all levels of abstraction from the learned hierarchical representation for ad-hoc retrieval.
- A duplex attention mechanism is designed to refine the multi-resolution so that an optimal matching of the query and document can be achieved. More specifically, the first mechanism determines a proper context, and the latter fine-tunes the representation by considering the interplay between the query and document.
- The proposed model significantly outperforms existing models for ad-hoc retrieval on major benchmark datasets.

The rest of the paper is organized as follows. First, we review recent advances in ad-hoc retrieval in Section 2. In Section 3 we describe the details of the proposed model. An empirical study to compare the proposed method to the existing approaches is conducted. The experiment and the result are reported in Section 4 and Section 5 respectively. Finally, we conclude the paper with some future research in Section 6.

2 RELATED WORK

The last few decades has seen increased adoption of machine learning for information retrieval (IR) tasks.

More recently, deep learning techniques (Lu and Li, 2013)(Hu et al., 2014)(Palangi et al., 2016)(Guo et al., 2016)(Hui et al., 2017) have successfully been applied to IR, or words document relevance ranking also known as *ad-hoc retrieval (Voorhees and Harman, 2005)*. In the ad-hoc retrieval task, the number of words in documents are generally greater than the number of words in queries and some might not even be in natural language form. This prevents aforementioned methods from other tasks that focus on pairs of short contents making them unsuitable for the task. Document ranking methods can be defined under the two titles: separation-oriented such as (Palangi et al., 2016) and interaction-oriented such as (McDonald et al., 2018). In the separation-oriented, a query and a document representations are generated separately. At the final step, interactions of these documents are getting calculated through dot-product whereby the result indicates relevance. In the interaction-oriented approach, specific encoding between the query and document pairs are induced which satisfies the exact matching as well as similarity matching that constitute the most important conditions for ad-hoc retrieval.

In the area of machine reading-style question answering (Rajpurkar et al., 2016) (Yang et al., 2015) (Dhingra et al., 2017) (Wang et al., 2007), the system needs to find the answer in the given corresponding document or context. The models have to combine information retrieval and machine reading. Note that we do not benchmark the quality of the extraction phase, therefore we do not study extracting the answer from the retrieved document, but compare the quality of retrieval methods, and the feasibility of learning specialized neural models for retrieval purposes. DrQA (Chen et al., 2017a) is built on top of two components: a Document Retriever and a Document Reader. The Document Retriever is a TF-IDF (Salton and McGill, 1986) retrieval system built upon Wikipedia corpus. Whereas, ConvRR (Cakaloglu and Xu, 2019) is a convolutional residual retrieval network that focuses on achieving better retrieval performance by employing a hard triplet mining. WordCnt, WgtWordCnt, PV, PV+Cnt, and CNN+Cnt are the models derived from the following study (Yang et al., 2015).Word Count method counts the number of non-stopwords in the question that also occur in the answer sentence, and Weighted Word Count re-weights the counts by the IDF values of the question words. PV represents the paragraph vector (Le and Mikolov, 2014) which is the similarity score between a question vector and document vector. CNN+Cnt is built on top of a bigram CNN (Yu et al., 2014) model with average pooling. PV+Cnt and CNN+Cnt are trained using a logistic regression classifier. QA-LSTM (Tan et al., 2016) is a

biLSTM based model where the final representations of question and document are taken by max or mean pooling over all the hidden vectors. ABCNN (Yin et al., 2016) is an attention-based convolution neural network model that employs an attention feature matrix to influence convolutions to optimize the task. RNN-POA (Chen et al., 2017b) is positional attention based RNN model that incorporates the positional content of the question words into the documents' attentive representations. SR^2: Simple Ranker-Reader, SR^3: Reinforced Ranker-Reader (Wang et al., 2018b) are proposed to improve the performance of the machine reading-style question answering tasks. They utilized the Apache Lucene-based search engine and a deep neural network ranker that re-ranks the documents retrieved by the search engine incorporated by a machine reader. The latter model is trained using reinforcement learning. The results derived from the models show that the neural network ranker can learn to rank the documents based on semantic similarity with the question. InferSent Ranker (Conneau et al., 2017)(Htut et al., 2018) is used to produce distributed representations for question and documents and, then, the input feature representation is built by concatenating the question representation, document representations, their difference, and their element-wise product. On receiving that input feature representation, the similarity score is calculated using a feed-forward neural network. Relation-Networks (RN) Ranker (Santoro et al., 2017)(Htut et al., 2018), further, calculates the relevance or local interactions between words in the question and paragraph. Thus, this model is built to interpret the relation between question-document pairs. Tree Edit Model (Heilman and Smith, 2010) is represented as sequences of tree transformations involving complex reordering phenomena and demonstrate a method for modeling pairs of semantically related contexts. They utilize a tree kernel in a greedy search routine to extract sequences of edits and use them in a logistic regression model to classify them. LSTM (Wang and Nyberg, 2015) uses a stacked bidirectional Long-Short Term Memory (BiLSTM) network to consecutively extract words from question and answer documents and then outputs their relevance scores. CNN (Severyn and Moschitti, 2015) is based on a convolutional neural network architecture for re-ranking pairs of short texts, that learns the optimal representation of document pairs and a similarity function to relate them. AP-LSTM (Tan et al., 2016) is developed considering hybrid models that handle the documents using both convolutional and recurrent neural networks incorporating attention mechanism to relate question and answer documents. AP-LSTM, AP-CNN (dos Santos et al., 2016) are based on two-way attention mechanisms for discriminative model (CNN, RNN, LSTM) training. AP allows the pooling layer to be aware of the input pair, in a way that information from the two can impact each other's representations. The model learns a similarity measure over projected n-Grams of the pair, and generate the attention representation for each input to lead the pooling. Self-LSTM, Multihop-Sequential-LSTM (Tran and Niederée, 2018) are developed to find the relationship between question and answer documents captured by attention. These models generate multiple representations that target different parts of the question. Additionally, they utilize sequential attention mechanism which uses context information for computing context-aware attention weights. The proposed *Multi-Resolution Neural Network* (MRNN) model belongs to this class of models of ad-hoc retrieval, but further incorporates the context of boosted interaction signals.

3 PROPOSED APPROACH

3.1 Overview

In this section, we introduce the proposed model called Multi-Resolution Neural Network as demonstrated in Figure 1. MRNN leverages the document relevance ranking and is composed of two major components: the initial component is responsible for generating n-Gram feature maps, the latter one matches those feature chains.

The model begins with a series of word inputs $e_1, e_2, e_3, ..., e_h$, that can establish a phrase, a sentence, a paragraph or a document. The model, then, builds *Multi-Resolution Feature Maps* through densely connected n-Gram blocks. To conscientiously extract most useful features for the target retrieval and matching tasks, a *Duplex Attention* component is introduced. The duplex attention component consists of two sub attention components that are called *Multi-Resolution n-Gram Attention* and *Document Aware Query Attention* respectively. The Multi-Resolution n-Gram Attention is responsible for reevaluating these Multi-Resolution feature maps while Document Aware Query Attention emphasizes matching those features between contextualized document and query to determine if they are similar or not. The final output is, then, used to improve the matching and retrieval performances.

3.2 Multi-Resolution Feature Maps

The first component called *Multi-Resolution Feature Maps* of the proposed model is shown in Figure 2. The component begins with a series of word inputs that

are initialized using one of the pre-trained embedding models. Let, $\mathbf{e}_i \in \mathbb{R}^w$ be the w-dimensional embedding vector of i-th word of the input text. Then, a matrix that is composed of all words of the input text is represented as follows:

$$\mathbf{E} = [\mathbf{e}_1, \mathbf{e}_2, \cdots, \mathbf{e}_h]_{h \times w} \quad (1)$$

where $\mathbf{E} \in \mathbb{R}^{h \times w}$ be the $h \times w$-dimensional matrix of the input text and h denotes the number of words in a given text. Embedded text, then, is fed to first n-Gram block of the component.

3.2.1 n-Gram Blocks

A n-Gram block is representing five cascaded operations: a convolution (*CONV*), a batch normalization (*BN*)(Ioffe and Szegedy, 2015), a parametric rectified linear unit (*PReLU*)(He et al., 2015), pooling operations (*POOLS*), and a scale unit *SU*. As the input text is represented with the matrix $\mathbf{E}$, the output representations of each *n-Gram* block within the component is denoted as below:

$$\mathbf{G}_n = [\mathbf{g}_n^1, \mathbf{g}_n^2, \cdots, \mathbf{g}_n^h]_{h \times s} \quad (2)$$

where s denotes the dimension of the transformed feature representation, n represents the index of an n-Gram block, and a total number of n-Gram blocks are labeled as N. Note that, while upstream n-Gram blocks are defined to emphasize the blocks that are located above the current block and close to the input layers of the network while the downstream blocks are located below the current block and close to the output layers of the network. In other words, feature maps derived from upstream blocks are associated with smaller n-Grams (unigram, bigram, and etc), on the other hand, downstream blocks are associated with larger n-Grams (6-grams, 7-grams, and etc) by reflecting on the feature maps derived from upstream n-Gram blocks.

Each n-Gram block utilizes $f_{gram}(\cdot,\cdot,\cdot)$ function to generate $\mathbf{G}_n \in \mathbb{R}^{h \times s}$ representations. $f_{gram}(\cdot,\cdot,\cdot)$ can be described as the below:

$$\mathbf{G}_n = f_{gram}(\mathbf{UB}, ws, s) \quad (3)$$

where $\mathbf{UB}$ denotes representations generated from upstream n-Gram blocks. ws, s represents the window size and the dimension of the transformed feature representation respectively. The definition of $\mathbf{UB}$ is conditioned to the index of the current n-Gram block.

$$\mathbf{UB} = \begin{cases} \mathbf{E}, & \text{if } n = 1 \\ [\mathbf{G}_1, \mathbf{G}_2, \cdots, \mathbf{G}_{n-1}], & \text{otherwise} \end{cases}$$

where $[\mathbf{G}_1, \mathbf{G}_2, \cdots, \mathbf{G}_{n-1}]$ shows the concatenation of the transformed representations derived from upstream n-Gram blocks ($1 \leq n-1$), in other words, densely connected upstream n-Gram blocks.

3.2.2 Densely Connected Blocks

Conventional models that are designed by placing convolution blocks consecutively, aim to extract hierarchical feature representations. Specifically, for text-oriented task scenarios, convolutions can be evaluated as to derive n-Gram features over a word sequence. Although traditional connections of convolution blocks extract hierarchical feature representations, they can not fulfill the requirements of natural languages for the following reasons:

- Traditional convolution based networks exploit the kernels of a constant size where a constant size window slides across all text to generate feature representations (Wang et al., 2017). This is called constant size n-Gram representations and it is not able to gather flexible size of n-Gram representations that are needed for better understanding of the text that depends on context, syntax, and semantics.
- In order to handle the extraction of flexible size n-Gram representations, one can employ the kernels with various window sizes, but another issue is fired up with such settings: What would be the right architecture of using different kernel sizes? In other terms, how much expansion does the network require for different kernel sizes to produce the best features? It would be a huge search space for the greedy search due to an exponential number of parameter combinations.
- Although the kernels with the variety of window sizes would still be seen as a better or an advanced architecture, it actually does not utilize the interplay between the representations derived from the different kernel size, since this type of approach consists of different independent parallel networks.

Therefore, we propose MRNN for ad-hoc retrieval, which leverages representations across all levels of abstractions in the learned hierarchical representation. The learned representation achieves a multi-resolution effect that can represent concepts and their relationship across all the levels of deep architecture. In order to consider all the resolutions (mixture of representations from adaptive n-Grams such that feature maps of words or short phrases from the upstream blocks affect the downstream blocks to compose feature maps for longer context) of the representations throughout the proposed network, we first employ dense connections between each n-Gram blocks inspired by ideas from computer vision (Huang et al., 2017) and text classifications (Kim et al., 2018) (Wang et al., 2018a).

$f_{gram}(\cdot,\cdot,\cdot)$ has input parameters of upstream block representations $\mathbf{UB}$, window size ws and, dimen-

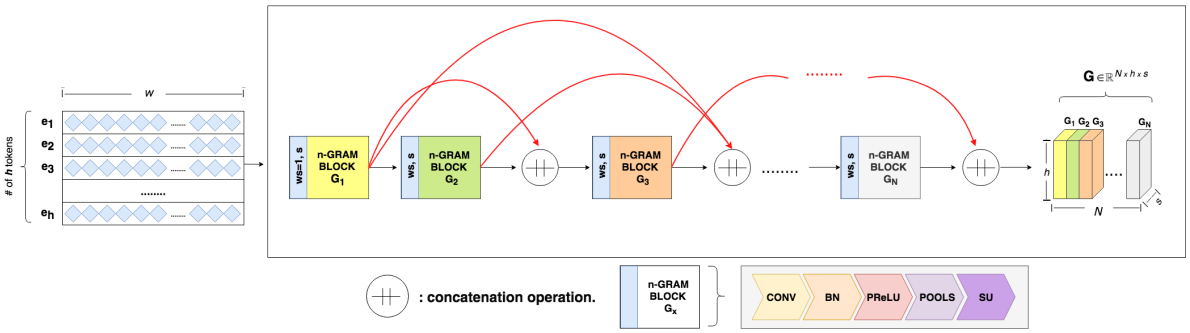

Figure 2: Multi-Resolution Feature Maps Component.

sion of the transformed representations s. Specifically, $f_{gram}(\cdot,\cdot,\cdot)$ computes the teachable weights $\mathbf{W}_n \in \mathbb{R}^{s\times ws\times s}$ by using the cascading aforementioned operations: *CONV*, *BN*, *PReLU*, *POOLS*, and *SU*. The teachable weight tensor $\mathbf{W}_n$ is composed of s filters where each of them has a matrix $\in \mathbb{R}^{ws\times s}$, convolving ws contiguous vectors. It is important to emphasize that in order to transform embedding dimension w to s, in the case where $n = 1$ or in other words $\mathbf{UB} = \mathbf{E}$, we employ s filters where each of them has a matrix $\in \mathbb{R}^{(ws=1)\times w}$ in the first phase. Additionally to prevent the output size of the feature maps being different after each n-Gram block, we use padding and pooling operations. We, further, apply a scalar unit to the feature map. That scalar sc is also teachable and it weights the feature map of the current n-th n-Gram block to decide how much the current block contributes to the downstream n-Gram blocks. Likewise, for the case where $\mathbf{UB} = [\mathbf{G}_1, \mathbf{G}_2, \cdots, \mathbf{G}_{n-1}]$ or, in other words, where $n > 1$, $f_{gram}(\cdot,\cdot,\cdot)$ computes the teachable weights $\mathbf{W}_n \in \mathbb{R}^{(n-1)\times s\times ws\times s}$ by using the same cascaded operations in the block. After considering all n-Gram blocks, the multi-resolution feature maps tensor is represented as follows: $\mathbf{G} = [\mathbf{G}_1, \mathbf{G}_2, \cdots, \mathbf{G}_N] \in \mathbb{R}^{N\times h\times s}$ and $\mathbf{G}$ consists of h matrices where each of them is representing the feature map matrix $\mathbf{G}_n$ from each n-Gram block.

3.3 Duplex Attention Component

Duplex attention component is one of the most important components of our proposed network. Since attention (Vaswani et al., 2017) evolves into an effective component within the neural network for extracting useful information, which achieves a remarkable result for many machine learning tasks, we compose this component with two sub attention components that are called *Multi-Resolution n-Gram Attention* and *Document Aware Query Attention* respectively. The multi-resolution feature maps tensors of each query(q)-document(d) pair are denoted as **Gq** - **Gd**. Hence, the Multi-Resolution n-Gram Attention is responsible for reevaluating **Gq**, and **Gd** feature maps while Document Aware Query Attention emphasizes matching those features between contextualized document and query to determine if they are similar or not.

3.3.1 Multi-Resolution n-Gram Attention

The multi-resolution feature maps tensor **G** (**Gq** or **Gd**) contains feature maps from all n-Gram blocks. More specifically, $\mathbf{Gq} = [\mathbf{Gq}_1, \mathbf{Gq}_2, \cdots, \mathbf{Gq}_N] \in \mathbb{R}^{N\times h_q\times s}$, and $\mathbf{Gd} = [\mathbf{Gd}_1, \mathbf{Gd}_2, \cdots, \mathbf{Gd}_N] \in \mathbb{R}^{N\times h_d\times s}$, the number of words in the query and the document are denoted as h_q and h_d respectively. Although the network has rich features at this step, some of those features still need to be pruned. In order to prune them conscientiously for the next component, we introduce a multi-resolution n-Gram attention component as shown in Figure 3. The multi-resolution n-Gram attention component has two consecutive functions called transformer $f_t(\cdot)$ and conductor $f_c(\cdot,\cdot)$ respectively. For the sake of simplicity, we explain the functions by taking the multi-resolution feature maps of the query (**Gq**) into consideration.

The transformer $f_t(\cdot)$ is a function that is formalized as below:

$$\mathbf{Tq} = f_t(\mathbf{Gq}) \tag{4}$$

where $\mathbf{Tq} \in \mathbb{R}^{N\times h_q}$ is a matrix of scalar adjusters. Since we know that $\mathbf{Gq}_n = [\mathbf{gq}_n^1, \mathbf{gq}_n^2, \cdots, \mathbf{gq}_n^{h_q}]_{h_q\times s}$ and each $\mathbf{gq}_n^i$ represents the s dimensional feature representation in the i-th location of a query at n-th n-Gram block. Particularly, for each $\mathbf{gq}_n^i$ feature representation, $f_t(\cdot)$ calculates the the scalar adjuster vector $\mathbf{tq}^i$:

$$\mathbf{tq}^i = [\sum_{j=1}^{s} \mathbf{gq}_1^i[j], \sum_{j=1}^{s} \mathbf{gq}_2^i[j], \cdots, \sum_{j=1}^{s} \mathbf{gq}_n^i[j]] \tag{5}$$

where $\mathbf{tq}^i \in \mathbb{R}^N$ is a N-dimensional vector, thus, $\mathbf{Tq} = [\mathbf{tq}^1, \mathbf{tq}^2, \cdots, \mathbf{tq}^{h_q}]$ is a matrix of scalar adjusters. The motivation behind this procedure is that the sum of all the values in the s-dimensional vector of $\mathbf{gq}_n^i$ is positioned by feature importance.

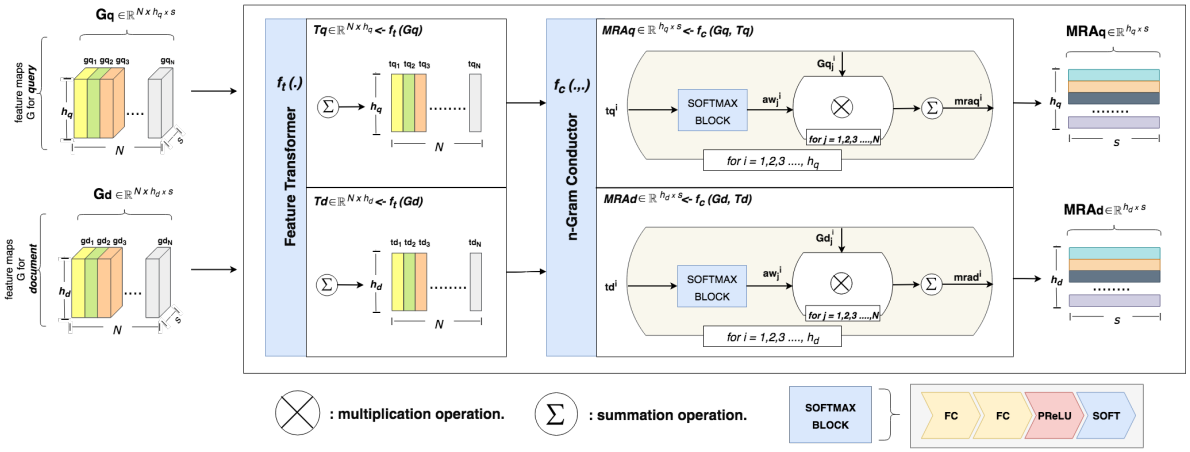

Figure 3: Multi-Resolution n-Gram Attention.

The scalar adjusters matrix **Tq** and the multi-resolution feature map tensor of query **Gq** are, further, passed to conductor function $f_c(\cdot,\cdot)$ to conduct the feature maps from variety of n-Gram scales by calculating attention weights via the softmax block. $f_c(\cdot,\cdot)$ is described as:

$$\mathbf{MRAq} = f_c(\mathbf{Gq}, \mathbf{Tq}) \tag{6}$$

where $\mathbf{MRAq} \in \mathbb{R}^{h_q \times s}$ is a matrix of multi-resolution n gram attention vectors. For each scalar vector $\mathbf{tq}^i$, $f_c(\cdot,\cdot)$, first, calculate the attention weights via softmax block represented as:

$$\mathbf{aw}^i = f(\mathbf{tq}^i) \tag{7}$$

where $f(\cdot)$ defines the softmax block that is a perceptron of the following cascaded operations: fully connected layers (FC), a parametric rectified linear unit ($PReLU$), and a softmax operation ($SOFT$). $\mathbf{aw} \in \mathbb{R}^N$ is an attention weights vector. Thus, representations of $\mathbf{tq}^i$ and $\mathbf{aw}^i$ can be shown as:

$$\mathbf{tq}^i = [tq_1^i, tq_2^i, \cdots, tq_N^i]$$
$$\mathbf{aw}^i = [aw_1^i, aw_2^i, \cdots, aw_N^i]$$

As next steps, $f_c(\cdot,\cdot)$ computes the final multi-resolution n gram attention vector using the following equations:

$$\mathbf{mraq}^i = \sum_{j=1}^{N} \mathbf{aw}_j^i \cdot \mathbf{Gq}_j^i \tag{8}$$

where $\mathbf{mraq}^i \in \mathbb{R}^s$ is a multi-resolution n gram attention vector of i-th position of a query. The final output representations of $f_c(\cdot,\cdot)$ is a multi-resolution n gram attention matrix **MRAq** that is evaluated as:

$$\mathbf{MRAq} = [\mathbf{mraq}^1, \mathbf{mraq}^2, \cdots, \mathbf{mraq}^{h_q}] \in \mathbb{R}^{h_q \times s}$$

Likewise, $\mathbf{MRAd} \in \mathbb{R}^{h_d \times s}$ is the multi-resolution n-Gram attention matrices for document. Both of matrices, then, are passed to next attention component called *Document Aware Query Attention*.

3.3.2 Document Aware Query Attention

In order to compute document aware query encoding using attention mechanism, we present *Document Aware Query Attention* component as illustrated in Figure 4. The component emphasizes matching features between the multi-resolution n-Gram attention matrices of query $\mathbf{MRAq} \in \mathbb{R}^{h_q \times s}$ and document $\mathbf{MRAd} \in \mathbb{R}^{h_d \times s}$ pair. The document aware query attention component has a function called encoder $f_e(\cdot,\cdot)$.

$f_e(\cdot,\cdot)$ is an encoder function that is formalized as below:

$$\mathbf{qa} = f_e(\mathbf{MRAq}, \mathbf{MRAd}) \tag{9}$$

where $\mathbf{qa} \in \mathbb{R}^{h_q}$ is a vector of document aware query encodings generated via attention weights. We, first, calculate a dot-product attention weights aw'_j for each position of **MRAd** relative to $\mathbf{mraq}^i$ via the softmax block as follows:

$$\mathbf{aw}'^i = f(\mathbf{mraq}^i \cdot \mathbf{MRAd}) \tag{10}$$

where $f(\cdot)$ defines the softmax block that is a perceptron with the same architecture (FCs, $PReLU$, $SOFT$). Note that, dot-products have a larger spectrum than the similar similarity functions, promising low entropy attention distributions. $\mathbf{aw}'^i \in \mathbb{R}^{h_d}$ is an attention weights vector. Thus, representation $\mathbf{aw}'^i$ can be shown as:

$$\mathbf{aw}'^i = [aw'^i_1, aw'^i_2, \cdots, aw'^i_{h_d}]$$

As a next step, we sum the document aware encodings of the h_d-locations, scaled by their attention weights, to create an attention-based representation $\mathbf{sae}^i$ of document representations **MRAd** from the query representation $\mathbf{mraq}_i$ formulated as below:

$$\mathbf{sae}^i = \sum_{j=1}^{h_d} \mathbf{aw}'^i_j \cdot \mathbf{mrad}^j \tag{11}$$

The element-wise distance (euclidean) between the attention-based document representation $\mathbf{sae}^i$ and

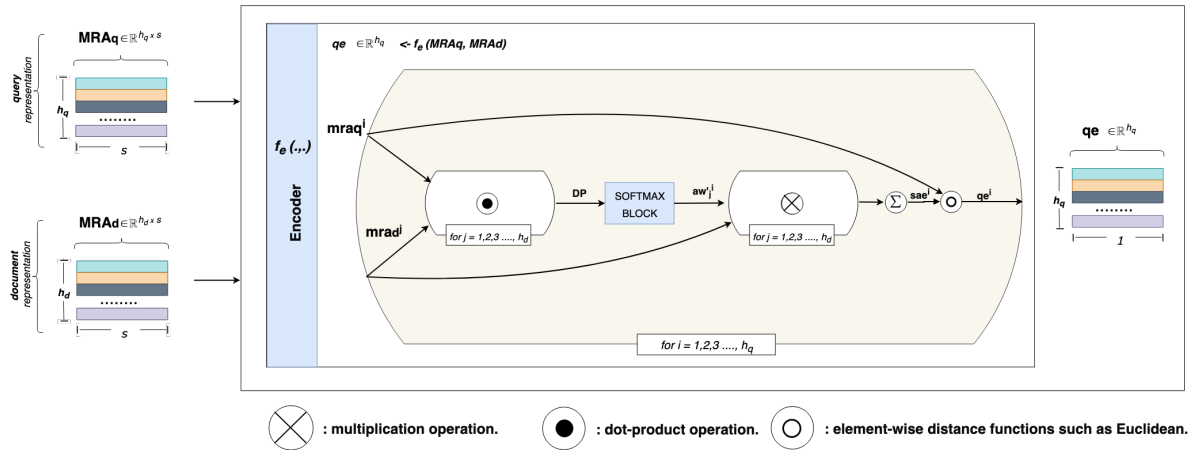

Figure 4: Document Aware Query Attention.

$\mathbf{mraq}^i$ is, further, calculated and adopted as document aware query encoding $qe^i \in \mathbb{R}^{h_q}$ that can be defined as a document aware query encoding of i-th position of a query. The final output representations of encoder $f_e(\cdot,\cdot)$ is a document aware query encoding attention vector **qe** that is evaluated as:

$$\mathbf{qe} = [qe^1, qe^2, \cdots, qe^{h_q}] \in \mathbb{R}^{h_q}$$

In other words, if the document includes more positions of $\mathbf{mrad}^j$ that are very much alike to the position of $\mathbf{mraq}^i$ in the query, the document aware query attention component indicates mostly those positions and, therefore $\mathbf{sae}^i$ will be close to $\mathbf{mraq}^i$.

As a final step, we aggregate these similarities qe^i via the *aggregate component* to define the distance of query-document pair as follows:

$$dist = \sum_{i=1}^{h_q} \mathbf{qe}^i \tag{12}$$

where $dist \in \mathbb{R}$ is the final distance descriptor between the query-document pair.

3.4 Distance Metric Loss Function

In order to train the proposed MRNN to perform well on retrieval and matching tasks, such that it also generalizes well on unseen data, we specifically utilized *triplet loss* function (Schroff et al., 2015) during the training period as shown in Figure 1 as distance metric loss. With this setup, the network is encouraged to reduce distances between positive pairs so that they are smaller than negative ones. A particular query Q would be a query anchor close in proximity to a document D^+ as the positive pair to the same question than to any document D^- as they are positive pairs to other questions. The key point of the L_{triplet} is to build the correct triplet structure which should meet the condition of the following equation:

$$\|Q, D^+\|^2 + m < \|Q, D^-\|^2$$

For each query, the document D^+ is selected as: $\arg\max_{D^+} \|Q, D^+\|^2$ and likewise the hardest document D^-: $\arg\min_{D^-} \|Q, D^-\|^2$ to form a triplet. This triplet selection strategy is called *hard triplets mining*.

Let $\mathbf{T} = (Q, D^+, D^-)$ be a triplet input. Given **T**, the proposed approach computes the distances between the positive and negative pairs via the proposed MRNN.

$$L_{\text{triplet}} = [\|Q, D^+\|^2 - \|Q, D^-\|^2 + m]^+ \tag{13}$$

where $m > 0$ is a scalar value called margin, and $\|\cdot,\cdot\|^2$ represents the distance score between two objects.

4 EXPERIMENTS

4.1 Datasets

In order to evaluate our proposed approach, we conducted extensive experiments on four datasets, including SQuAD (Rajpurkar et al., 2016), WikiQA (Yang et al., 2015), QUASAR (Dhingra et al., 2017), and TrecQA (Wang et al., 2007).

4.1.1 SQuAD

The Stanford Question Answering Dataset (SQuAD) (Rajpurkar et al., 2016) is a large reading comprehension dataset that is built with $100,000+$ questions. Each of these questions is composed by crowd workers on a set of Wikipedia documents where the answer to each question is a segment of text from the corresponding reading passage. In other words, the consolidation of retrieval and extraction tasks are aimed at measuring the success of the proposed systems.

4.1.2 WikiQA

The Wikipedia open-domain Question Answering (WikiQA) (Yang et al., 2015) dataset is collected using Bing query logs. For each question, clicked Wikipedia pages (issued by at least 5 unique users) and used sentences in the summary section of Wikipedia page as the candidates, is further marked on a crowdsourcing platform. Note that we excluded questions that have no candidates. Based on training, dev and test subsets, 1,242 questions are used in the experiment.

4.1.3 QUASAR

The Question Answering by Search And Reading (QUASAR) is a large-scale dataset consisting of QUASAR-S and QUASAR-T. Each of these datasets is built to focus on evaluating systems devised to understand a natural language query, a large corpus of texts and to extract an answer to the question from the corpus. Similar to SQuAD QUASAR is primarily used to measure the success of the proposed systems for ad-hoc retrieval and extraction tasks. Specifically, QUASAR-S comprises 37,012 fill-in-the-gaps questions that are collected from the popular website Stack Overflow using entity tags. Since our research is not to address the fill-in-the-gaps questions, we want to pay attention to the QUASAR-T dataset that fulfills the requirements of our focused retrieval task. The QUASAR-T dataset contains 43,012 open-domain questions collected from various internet sources. The candidate documents for each question in this dataset are retrieved from an Apache Lucene based search engine built on top of the ClueWeb09 dataset (Callan et al., 2009).

4.1.4 TrecQA

Text Retrieval Conference Question Answering (TrecQA)(Wang et al., 2007) is a popular benchmark dataset for question answering. TrecQA dataset is based on QA track (8-13) of TREC. The dataset consists of factoid questions, each of which has a single sentence as a candidate answer. In order to make the comparison parallel to the previous works, we pursue the same strategy they applied where all questions with only positive or negative answers are excluded. In total, we end up having 1,295 questions within all training, dev and test subsets of TrecQA.

The number of queries in each dataset including their subsets is listed in Table 1.

Table 1: Datasets Statistics: Number of queries in each train, validation, and test subsets.

DATASET	TRAIN	VALID.	TEST	TOTAL
SQUAD	87,599	10,570	HIDDEN	98,169+
WIKIQA	873	126	243	1,242
QUASAR-T	37,012	3,000	3,000	43,012
TRECQA	1,162	65	68	1,295

4.2 Performance Measure

The matching and retrieval tasks aim to improve the *recall@k*, the Mean Reciprocal Rank (*MRR*) and Mean Average Precision (*MAP*). The *recall@k* score is calculated by selecting the correct pair among all candidates. Basically, *recall@k* is defined as the number of correct documents as listed within top-*k* returns out all possible documents. Likewise, *MRR* and *MAP* metrics are also commonly used in information retrieval and question answering researches (Manning et al., 2008).

4.3 Implementation

4.3.1 Input

We adopt the multi-resolution word embedding (Cakaloglu and Xu, 2019) (Cakaloglu et al., 2022) using Bert(Devlin et al., 2018), ELMo(Peters et al., 2018), FastText(Mikolov et al., 2018) for each question and document in datasets. We configure the multi-resolution word embedding stated in (Cakaloglu and Xu, 2019): $f_{mixture}(\cdot,\cdot,\cdot,\cdot)$ and $f_{ensemble}(\cdot,\cdot)$ configurations are shown in Table 2 and Table 3 respectively.

Table 2: $f_{mixture}(\cdot,\cdot,\cdot)$ configuration of the multi-resolution word embedding.

E	w_{idf}	m	f_{mix}	OUT
BERT	0	$[\frac{1}{4}, \frac{1}{4}, \frac{1}{4}, \frac{1}{4}, ..., 0]$	*conc.*	$\mathbf{x}^1$
ELMO	1	[0, 0, 1]	*sum*	$\mathbf{x}^2$
FASTTEXT	1	[1]	*sum*	$\mathbf{x}^3$

Table 3: $f_{ensemble}(\cdot)$ configuration of the multi-resolution word embedding.

X'	u	$f_{ensemble}$
$\{\mathbf{x}^1, \mathbf{x}^2, \mathbf{x}^3\}$	$[\frac{1}{3}, \frac{1}{3}, \frac{1}{3}]$	*concat.*

4.3.2 MRNN Training Configuration

The proposed MRNN is implemented with Tensorflow 1.8+ by (Abadi et al., 2015) and trained on NVIDIA Tesla K40c GPUs. Specifically, the network is trained

using ADAM optimizer (Kingma and Ba, 2014) with a batch size of 512. The learning rate is set to 10^{-4}. Additionally, the weight decay is set to 10^{-3} to tackle over-fitting. The triplet loss is, then, chosen as an objective function with different margins for each of the datasets. ($m = 1$: SQuAD, $m = 0.8$: QUASAR-T, $m = 0.5$: {WikiQA ,TrecQA}). We configured 6 n-Gram blocks ($N = 6$) for SQuAD and QUASAR-T datasets, and 4 n-Gram blocks ($N = 4$) for WikiQA and TrecQA datasets. Last but not least, window size and transformed feature representation dimension are set to $ws = 3$, $s = 1024$ respectively.

5 RESULTS

We compare our approach with different models proposed by other researchers for each dataset using their evaluation measures and test subsets.

5.1 SQuAD

The *recall*@5 result is calculated for SQuAD in order to compare with the document retrieval component of the multi-layer recurrent neural network (Chen et al., 2017a) (DrQA) and the convolutional residual retrieval network (ConvRR) (Cakaloglu and Xu, 2019)(Cakaloglu et al., 2018). The comparisons are shown in Table 4.

5.2 WikiQA

WordCnt, WgtWordCnt, and CNN-Cnt are the models derived from the following initial study (Yang et al., 2015). On top of the baseline models, the Paragraph Vector (PV) and PV + Cnt models (tau Yih et al., 2013) are taken into consideration. We, further, consider even more advanced models: QA-LSTM (Tan et al., 2016), Self-LSTM, Multihop-Sequential-LSTM (Tran and Niederée, 2018), ABCNN (Yin et al., 2016), Rank MP-CNN (Rao et al., 2016), RNN-POA (Chen et al., 2017b). In order to compare the proposed model with the aforementioned models for the WikiQA dataset, we calculate the Mean Reciprocal Rank (*MRR*) and Mean Average Precision (*MAP*) metrics and all the results are presented in Table 5.

5.3 QUASAR

BM25 (Robertson and Zaragoza, 2009)(Htut et al., 2018), SR^2: Simple Ranker-Reader, SR^3: Reinforced Ranker-Reader (Wang et al., 2018b), InferSent Ranker (Conneau et al., 2017)(Htut et al., 2018), convolutional residual retrieval network (ConvRR) (Cakaloglu and Xu, 2019), and Relation-Networks (RN) Ranker (Santoro et al., 2017)(Htut et al., 2018) are the models that are evaluated using the *recall*@1, *recall*@3, and *recall*@5. The comparisons are listed in Table 6.

5.4 TrecQA

We compute the Mean Reciprocal Rank (*MRR*) and Mean Average Precision (*MAP*) metrics for the proposed model as well as following models: Tree Edit Model (Heilman and Smith, 2010), LSTM (Wang and Nyberg, 2015), CNN (Severyn and Moschitti, 2015), AP-LSTM (Tan et al., 2016), AP-CNN (dos Santos et al., 2016), RNN-POA (Chen et al., 2017b), and Self-LSTM, Multihop-Sequential-LSTM (Tran and Niederée, 2018). The results are stated in Table 7.

5.5 Evaluation

The result on all benchmark datasets shows that the proposed MRNN clearly outperforms existing models. Relation-Networks Ranker is the only model that achieved a slightly better result for recall@3 and recall@5 on QUASAR-T dataset. The proposed MRNN model however outperforms Relation-Networks Ranker on recall@1.

Table 4: Performances on SQUAD. *recall@k* retrieved documents using the baseline models and the proposed model.

MODEL	@5
CONVRR	75.6
DRQA DOCUMENT-RETRIEVAL	77.8
MRNN	**80.4**

Table 5: Performances on WikiQA. The baseline models and the proposed model are listed based on the results derived from *MAP* and *MRR* metrics.

MODEL	MAP	MRR
WORDCOUNT	0.4891	0.4924
WGTWORDCNT	0.5099	0.5132
PV	0.511	0.516
PV + CNT	0.599	0.609
CNN + CNT	0.652	0.6652
QA-LSTM	0.654	0.665
AP-LSTM	0.670	0.684
AP-CNN	0.689	0.696
SELF-LSTM	0.693	0.704
ABCNN	0.692	0.71
RANK MP-CNN	0.701	0.718
RNN-POA	0.721	0.731
MULTIHOP-SEQUENTIAL-LSTM	0.722	0.738
MRNN	**0.731**	**0.745**

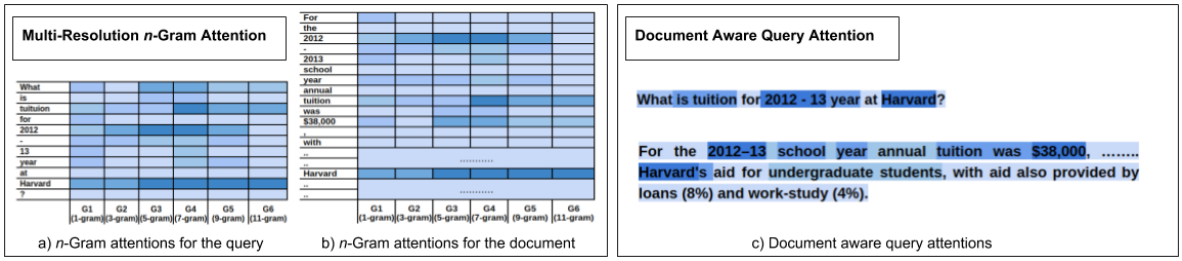

Figure 5: For the sample query and document extracted from SQuAD, the visualization of their attention weights that are fired in the sub attention components of Duplex Attention.

Table 6: Performances on QUASAR-T. *recall@k* retrieved documents using the baseline models and the proposed model.

MODEL	@1	@3	@5
BM25	19.7	36.3	44.3
SR^2: SIMPLE R-R	28.8	46.4	54.9
INFERSENT RANKER	36.1	52.8	56.7
SR^3: REINFORCED R-R	40.3	51.3	54.5
CONVRR	50.67	63.1	67.4
RELATION-NETWORKS R	51.4	**68.2**	**70.3**
MRNN	**52.8**	67.7	69.9

Table 7: Performances on TrecQA. The baseline models and the proposed model are listed based on the results derived from *MAP* and *MRR* metrics.

MODEL	MAP	MRR
TREE EDIT MODEL	0.609	0.692
LSTM	0.713	0.791
CNN	0.746	0.808
AP-LSTM	0.753	0.830
AP-CNN	0.753	0.851
SELF-LSTM	0.759	0.830
RNN-POA	0.781	0.851
MULTIHOP-SEQUENTIAL-LSTM	0.813	0.893
MRNN	**0.822**	**0.898**

5.6 Visualization and Analysis

The attention visualizations of the duplex attention component for the sample query and corresponding document extracted from the SQuAD dataset are shown in Figure 5. When window-size is set to 3 ($ws = 3$), and the number of n-Gram blocks is set to 6 ($N = 6$), then each row of Figure 5, a) and b) indicate an attention weight distribution over $\mathbf{gq}_n^i$ as well as $\mathbf{gd}_n^i$ and i is the position of each words in the matrices of the query and document. Hence, $\mathbf{Gq}_n$ and $\mathbf{Gd}_n$ corresponds to feature maps of (*2n-1*)-gram, e.g., $\mathbf{Gq}_1$: 1-gram, $\mathbf{Gq}_2$:3-gram, ... etc. The proposed MRNN puts more emphasizes on some segments of the query and the parts of the document as in the multi-resolution n-Gram attention. In the document aware query attention, MRNN gives more attention to additional segments of the question and also some other parts of the document that have crucial interactions, as shown in Figure 5, c).

6 CONCLUSION

Ad-hoc retrieval is an important task for question answering and information retrieval. This paper proposes a new multi-resolution neural network for ad-hoc retrieval, which is the first model that leverages the strength of representations of different abstract levels in the learned hierarchical representation. The proposed model incorporated with a new duplex attention mechanism can significantly improve the performance of ad-hoc retrieval as demonstrated by the superior results in comparison to other existing methods on major benchmark datasets. More specifically our study shows that MRNN with the duplex attention improves gain on existing model performances over different evaluation metrics and benchmark datasets by a range of 0.5% to 1.5% .In the future, we want to apply the proposed model to other areas including pattern recognition and computer vision.

REFERENCES

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mané, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viégas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y., and Zheng, X. (2015). TensorFlow: Large-scale machine learning on heterogeneous systems. Software available from tensorflow.org.

Cakaloglu, T., Szegedy, C., and Xu, X. (2018). Text embed-

dings for retrieval from a large knowledge base. *arXiv preprint arXiv:1810.10176*.

Cakaloglu, T. and Xu, X. (2019). A multi-resolution word embedding for document retrieval from large unstructured knowledge bases. *arXiv preprint*.

Cakaloglu, T., Xu, X., and Raghavan, R. (2022). Emboost: Embedding boosting to learn multilevel abstract text representation for document retrieval. In Rocha, A. P., Steels, L., and van den Herik, H. J., editors, *Proceedings of the 14th International Conference on Agents and Artificial Intelligence, ICAART 2022, Volume 3, Online Streaming, February 3-5, 2022*, pages 352–360. SCITEPRESS.

Callan, J., Hoy, M., Yoo, C., and Zhao, L. (2009). Clueweb09 data set.

Chen, D., Fisch, A., Weston, J., and Bordes, A. (2017a). Reading wikipedia to answer open-domain questions. In *ACL*.

Chen, Q., Hu, Q., Huang, X., He, L., and An, W. (2017b). Enhancing recurrent neural networks with positional attention for question answering. In *SIGIR*.

Conneau, A., Kiela, D., Schwenk, H., Barrault, L., and Bordes, A. (2017). Supervised learning of universal sentence representations from natural language inference data. In *EMNLP*.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Dhingra, B., Mazaitis, K., and Cohen, W. W. (2017). Quasar: Datasets for question answering by search and reading. *arXiv preprint arXiv:1707.03904*.

dos Santos, C. N., Tan, M., Xiang, B., and Zhou, B. (2016). Attentive pooling networks. *CoRR*, abs/1602.03609.

Guo, J., Fan, Y., Ai, Q., and Croft, W. B. (2016). A deep relevance matching model for ad-hoc retrieval. In *CIKM*.

He, K., Zhang, X., Ren, S., and Sun, J. (2015). Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 1026–1034.

Heilman, M. and Smith, N. A. (2010). Tree edit models for recognizing textual entailments, paraphrases, and answers to questions. In *HLT-NAACL*.

Htut, P. M., Bowman, S. R., and Cho, K. (2018). Training a ranking function for open-domain question answering. In *NAACL-HLT*.

Hu, B., Lu, Z., Li, H., and Chen, Q. (2014). Convolutional neural network architectures for matching natural language sentences. In *NIPS*.

Huang, G., Liu, Z., v. d. Maaten, L., and Weinberger, K. Q. (2017). Densely connected convolutional networks. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2261–2269.

Hui, K., Yates, A., Berberich, K., and de Melo, G. (2017). Pacrr: A position-aware neural ir model for relevance matching. In *EMNLP*.

Ioffe, S. and Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proceedings of the 32Nd International Conference on International Conference on Machine Learning - Volume 37*, ICML'15, pages 448–456. JMLR.org.

Kim, S., Hong, J.-H., Kang, I., and Kwak, N. (2018). Semantic sentence matching with densely-connected recurrent and co-attentive information. *CoRR*, abs/1805.11360.

Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Le, Q. V. and Mikolov, T. (2014). Distributed representations of sentences and documents. In *ICML*.

Lu, Z. and Li, H. (2013). A deep architecture for matching short texts. In *NIPS 2013*.

Manning, C. D., Raghavan, P., and Schütze, H. (2008). *Introduction to Information Retrieval*. Cambridge University Press.

McDonald, R., Ding, Y., and Androutsopoulos, I. (2018). Deep relevance ranking using enhanced document-query interactions. In *EMNLP*.

Mikolov, T., Grave, E., Bojanowski, P., Puhrsch, C., and Joulin, A. (2018). Advances in pre-training distributed word representations. In *Proceedings of the International Conference on Language Resources and Evaluation (LREC 2018)*.

Palangi, H., Deng, L., Shen, Y., Gao, J., He, X., Chen, J., Song, X., and Ward, R. K. (2016). Deep sentence embedding using long short-term memory networks: Analysis and application to information retrieval. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 24:694–707.

Peters, M., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., and Zettlemoyer, L. (2018). Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237.

Rajpurkar, P., Zhang, J., Lopyrev, K., and Liang, P. (2016). Squad: 100, 000+ questions for machine comprehension of text. *arXiv preprint arXiv:1606.05250*.

Rao, J., He, H., and Lin, J. J. (2016). Noise-contrastive estimation for answer selection with deep neural networks. In *CIKM*.

Robertson, S. and Zaragoza, H. (2009). The probabilistic relevance framework: Bm25 and beyond. *Foundations and Trends in Information Retrieval*, 3:333–389.

Salton, G. and McGill, M. J. (1986). Introduction to modern information retrieval.

Santoro, A., Raposo, D., Barrett, D. G. T., Malinowski, M., Pascanu, R., Battaglia, P. W., and Lillicrap, T. P. (2017). A simple neural network module for relational reasoning. In *NIPS*.

Schroff, F., Kalenichenko, D., and Philbin, J. (2015). Facenet: A unified embedding for face recognition and clustering. *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 815–823.

Severyn, A. and Moschitti, A. (2015). Learning to rank short text pairs with convolutional deep neural networks. In *SIGIR*.

Tan, M., dos Santos, C. N., Xiang, B., and Zhou, B. (2016). Improved representation learning for question answer matching. In *ACL*.

tau Yih, W., Chang, M.-W., Meek, C., and Pastusiak, A. (2013). Question answering using enhanced lexical semantic models. In *ACL*.

Tran, N. K. and Niederée, C. (2018). Multihop attention networks for question answer matching. In *SIGIR*.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L. u., and Polosukhin, I. (2017). Attention is all you need. In *Advances in Neural Information Processing Systems 30*, pages 5998–6008.

Voorhees, E. M. and Harman, D. K. (2005). *TREC: Experiment and Evaluation in Information Retrieval (Digital Libraries and Electronic Publishing)*. The MIT Press.

Wang, D. and Nyberg, E. (2015). A long short-term memory model for answer sentence selection in question answering. In *ACL*.

Wang, M., Smith, N. A., and Mitamura, T. (2007). What is the jeopardy model? a quasi-synchronous grammar for qa. In *EMNLP-CoNLL*.

Wang, S., Huang, M., and Deng, Z. (2018a). Densely connected cnn with multi-scale feature attention for text classification. In *IJCAI*.

Wang, S., Yu, M., Guo, X., Wang, Z., Klinger, T., Zhang, W., Chang, S., Tesauro, G., Zhou, B., and Jiang, J. (2018b). R$\hat{3}$: Reinforced ranker-reader for open-domain question answering. In *AAAI 2018*.

Wang, Z., Wang, Z., Zhang, D., and Yan, J. (2017). Combining knowledge with deep convolutional neural networks for short text classification. In *IJCAI*.

Yang, Y., tau Yih, W., and Meek, C. (2015). Wikiqa: A challenge dataset for open-domain question answering. In *EMNLP*.

Yin, W., Schütze, H., Xiang, B., and Zhou, B. (2016). Abcnn: Attention-based convolutional neural network for modeling sentence pairs. *Transactions of the Association for Computational Linguistics*, 4:259–272.

Yu, L., Hermann, K. M., Blunsom, P., and Pulman, S. G. (2014). Deep learning for answer sentence selection. *CoRR*, abs/1412.1632.