Multi-Cast Attention Networks for Retrieval-based Question Answering and Response Prediction

Yi Tay  
Nanyang Technological University  
Singapore  
ytay017@e.ntu.edu.sg

Luu Anh Tuan  
Institute for Infocomm Research  
Singapore  
at.luu@i2r.a-star.edu.sg

Siu Cheung Hui  
Nanyang Technological University  
Singapore  
asschui@ntu.edu.sg

ABSTRACT

Attention is typically used to select informative sub-phrases that are used for prediction. This paper investigates the novel use of attention as a form of feature augmentation, i.e., casted attention. We propose Multi-Cast Attention Networks (MCAN), a new attention mechanism and general model architecture for a potpourri of ranking tasks in the conversational modeling and question answering domains. Our approach performs a series of soft attention operations, each time casting a scalar feature upon the inner word embeddings. The key idea is to provide a real-valued hint (feature) to a subsequent encoder layer and is targeted at improving the representation learning process. There are several advantages to this design, e.g., it allows an arbitrary number of attention mechanisms to be casted, allowing for multiple attention types (e.g., co-attention, intra-attention) and attention variants (e.g., alignment-pooling, max-pooling, mean-pooling) to be executed simultaneously. This not only eliminates the costly need to tune the nature of the co-attention layer, but also provides greater extents of explainability to practitioners. Via extensive experiments on four well-known benchmark datasets, we show that MCAN achieves state-of-the-art performance. On the Ubuntu Dialogue Corpus, MCAN outperforms existing state-of-the-art models by 9%. MCAN also achieves the best performing score to date on the well-studied TrecQA dataset.

KEYWORDS

Deep Learning; Information Retrieval; Question Answering; Conversation Modeling

ACM Reference Format:

Yi Tay, Luu Anh Tuan, and Siu Cheung Hui. 2018. Multi-Cast Attention Networks for Retrieval-based Question Answering and Response Prediction. In Proceedings of ACM Conference (Conference'17). ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/nnnnnn.nn

1 INTRODUCTION

Modeling textual relevance between document query pairs lives at the heart of information retrieval (IR) research. Intuitively, this enables a wide assortment of real life applications, ranging from standard web search to automated chatbots. The key idea is that these systems learn a scoring function between document-query pairs, providing a ranked list of candidates as an output. A considerable fraction of such IR systems are focused on short textual documents, e.g., answering facts based questions or selecting the best response in the context of a chat-based system. The application of retrieval-based response and question answering (QA) systems is overall versatile, potentially serving as a powerful standalone domain-specific system or a crucial component in larger, general purpose chat systems such as Alexa. This paper presents a universal neural ranking model for such tasks.

Neural networks (or deep learning) have garnered considerable attention for retrieval-based systems [8, 27, 29, 39, 52]. Notably, the dominant state-of-the-art systems for many benchmarks are now neural models, almost completely dispensing with traditional feature engineering techniques altogether. In these systems, convolutional or recurrent networks are empowered with recent techniques such as neural attention [1, 25, 39], achieving very competitive results on standard benchmarks. The key idea of attention is to extract only the most relevant information that is useful for prediction. In the context of textual data, attention learns to weight words and sub-phrases within documents based on how important they are. In the same vein, co-attention mechanisms [5, 28, 50, 54] are a form of attention mechanisms that learn joint pairwise attentions, with respect to both document and query.

Attention is traditionally used and commonly imagined as a feature extractor. Its behavior can be thought of as a dynamic form of pooling as it learns to select and compose different words to form the final document representation. This paper re-imagines attention as a form of feature augmentation method. Attention is casted with the purpose of not compositional learning or pooling but to provide hints for subsequent layers. To the best of our knowledge, this is a new way to exploit attention in neural ranking models. We begin by describing not only its advantages but also how it handles the weaknesses of existing models designed today.

Typically, attention is applied once to a sentence [25, 39]. A final representation is learned, and then passed to prediction layers. In the context of handling sequence pairs, co-attention is applied and a final representation for each sentence is learned [5, 28, 54]. An obvious drawback which applies to many existing models is that they are generally restricted to one attention variant. In the case where one or more attention calls are used (e.g., co-attention and intra-attention, etc.), concatenation is generally used to fuse representations [20, 28]. Unfortunately, this incurs cost in subsequent layers by doubling the representation size per call.

The rationale for desiring more than one attention call is intuitive. In [20, 28], Co-Attention and Intra-Attention are both used because each provides a different view of the document pair, learning high
quality representations that could be used for prediction. Hence, this can significantly improve performance. Moreover, Co-Attention also comes in different flavors and can either be used with extractive max-mean pooling [5, 54] or alignment-based pooling [3, 20, 28]. Each co-attention type produces different document representations. In max-pooling, signals are extracted based on a word’s largest contribution to the other text sequence. Mean-pooling calculates its contribution to the overall sentence. Alignment-pooling is another flavor of co-attention, which aligns semantically similar sub-phrases together. As such, different pooling operators provide a different view of sentence pairs. This is often tuned as a hyperparameter, i.e., performing architectural engineering to find the best variation that works best on each problem domain and dataset.

Our approach is targeted at serving two important purposes - (1) It removes the need for architectural engineering of this component by enabling attention to be called for an arbitrary k times with hardly any consequence and (2) concurrently it improves performance by modeling multiple views via multiple attention calls. As such, our method is in similar spirit to multi-headed attention, albeit with the casted attention features are interpretable and are capable of learning (1) a neural adaptation of word overlap and (2) a differentiation of evidence and anti-evidence words/patterns.

2 MULTI-CAST ATTENTION NETWORKS
In this section, we describe our proposed MCAN model. The inputs to our model are two text sequences which we denote as query q and document d. In our problem, query-document can be generalized to different problem domains such as question-answering or message-response prediction. Figure 1 illustrates the overall model architecture for question-answer retrieval.

2.1 Input Encoder
The document and query inputs are passed in as one-hot encoded vectors. A word embedding layer parameterized by \( W_e \in \mathbb{R}^{d \times |V|} \) converts each word to a dense word representation \( w \in \mathbb{R}^d \), \( V \) is the set of all words in the vocabulary.

2.1.1 Highway Encoder. Each word embedding is passed through a highway encoder layer. Highway networks [31] are gated non-linear transform layers which control information flow to subsequent layers. Many works adopt a projection layer that is trained in place of the raw word embeddings. Not only does this save computation cost but also reduces the number of trainable parameters. Our work extends this projection layer to use a highway encoder. The intuition for doing so is simple, i.e., highway encoders can be interpreted as data-driven word filters. As such, we can imagine them to parametrically learn which words have an inclination to be important and not important to the task at hand. For example, filtering stop words and words that usually do not contribute much to the
prediction. Similar to recurrent models that are gated in nature, this highway encoder layer controls how much information (of each word) is flowed to the subsequent layers.

Let \( H(\cdot) \) and \( T(\cdot) \) be single layered affine transforms with ReLU and sigmoid activation functions respectively. A single highway network layer is defined as:

\[
y = H(x, W_H) \cdot T(x, W_T) + (1 - T(x, W_T)) \cdot x
\]

where \( W_H, W_T \in \mathbb{R}^{r \times d} \). Notably, the dimensions of the affine transform might be different from the size of the input vector. In this case, an additional nonlinear transform is used to project \( x \) to the same dimensionality.

2.2 Co-Attention

Co-Attention [50] is a pairwise attention mechanism that enables attending to text sequence pairs jointly. In this section, we introduce four variants of attention, i.e., (1) max-pooling, (2) mean-pooling, (3) alignment-pooling, and finally (4) intra-attention (or self-attention).

The first step in co-attention is to learn an affinity (or similarity) matrix between each word across both sequences. Following Parikh et al. [20], we adopt the following formulation for learning the affinity matrix.

\[
s_{ij} = F(q_i)^T F(d_j)
\]

where \( F(\cdot) \) is a function such as a multi-layered perceptron (MLP). Alternate forms of co-attention are also possible such as \( s_{ij} = q_i^T M d_j \) and \( s_{ij} = F(q_i; d_j) \).

2.2.1 Extractive Pooling. The most common variant of extractive co-attention is the max-pooling co-attention, which attends to each word based on its maximum influence it has on the other text sequence.

\[
q' = \text{Soft}(\text{max}(s))^\top q \quad \text{and} \quad d' = \text{Soft}(\text{max}(s))^\top d
\]

where \( q', d' \) are the co-attentional representations of \( q \) and \( d \) respectively. Soft(\( \cdot \)) is the Softmax operator. Alternatively, the mean row and column-wise pooling of matrix \( s \) can also be used:

\[
q' = \text{Soft}(\text{mean}(s))^\top q \quad \text{and} \quad d' = \text{Soft}(\text{mean}(s))^\top d
\]

However, each pooling operator has different impacts and can be intuitively understood as follows: max-pooling selects each word based on its maximum importance of all words in the other text. Mean-pooling is a more wholesome comparison, paying attention to a word based on its overall influence on the other text. This is usually dataset-dependent, regarded as a hyperparameter and is tuned to see which performs best on the held out set.

2.2.2 Alignment-Pooling. Soft alignment-based pooling has also been utilized for learning co-attentive representations [20]. However, the key difference with soft alignment is that it realigns sequence pairs while standard co-attention simply learns to weight and score important words. The co-attentive representations are then learned as follows:

\[
d'_j := \sum_{j=1}^{\ell_q} \frac{\exp(s_{ij})}{\sum_{k=1}^{\ell_q} \exp(s_{jk})} q_j \quad \text{and} \quad d'_i := \sum_{i=1}^{\ell_d} \frac{\exp(s_{ij})}{\sum_{k=1}^{\ell_d} \exp(s_{kj})} d_i
\]

where \( d'_j \) is the sub-phrase in \( q \) that is softly aligned to \( d_i \). Intuitively, \( d'_i \) is a weighted sum across \( \{q_j\}_{j=1}^{\ell_q} \), selecting the most relevant parts of \( q \) to represent \( d_i \).

2.2.3 Intra-Attention. Intra-Attention, or Self-Attention was recently proposed to learn representations that are aware of long-term dependencies. This is often formulated as an co-attention (or alignment) operation with respect to itself. In this case, we apply intra-attention to both document and query independently. For notational simplicity, we refer to them as \( x \) instead of \( q \) or \( d \) here. The Intra-Attention function is defined as:

\[
x'_j := \sum_{j=1}^{\ell} \frac{\exp(s_{ij})}{\sum_{k=1}^{\ell} \exp(s_{ik})} x_j
\]

where \( x'_j \) is the intra-attentional representation of \( x_j \).

2.3 Multi-Cast Attention

At this point, it is easy to make several observations. Firstly, each attention mechanism provides a different flavor to the model. Secondly, attention is used to alter the original representation either by re-weighting or realigning. As such, most neural architectures only make use of one type of co-attention or alignment function [5, 20]. However, this requires the right model architecture to be tuned and potentially missing out from the benefits brought by using multiple variations of co-attention mechanism. As such, our work casts each attention operation as a word-level feature.

2.3.1 Casted Attention. Let \( x \) be either \( q \) or \( d \) and \( \hat{x} \) is the representation\(^4\) of \( x \) after applying co-attention or soft attention alignment. The attention features for the co-attention operators are:

\[
f_c = F_c(\hat{x} \odot x) \quad \text{(7)}
\]

\[
f_m = F_c(\hat{x} \odot x) \quad \text{(8)}
\]

\[
f_k = F_k(\hat{x} - x) \quad \text{(9)}
\]

where \( \odot \) is the Hadamard product and \([\cdot; \cdot]\) is the concatenation operator. \( F_c(\cdot) \) is a compression function used to reduce features to a scalar. Intuitively, what is achieved here is that we are modeling the influence of co-attention by comparing representations before and after co-attention. For soft-attention alignment, a critical note here is that \( x \) and \( \hat{x} \) (though of equal lengths) have ‘exchanged’ semantics. In other words, in the case of \( q, q' \) actually contains the aligned representation of \( d \). Finally, the usage of multiple comparison operators (subtractive, concatenation and multiplicative operators) is to capture multiple perspectives and is inspired by the ESIM model [3].

2.3.2 Compression Function. This section defines \( F_c(\cdot) \) the compression function used. The rationale for compression is simple and intuitive - we do not want to bloat subsequent layers with a high dimensional vector which consequently incurs parameter costs in subsequent layers. We investigate the usage of three compression functions, which are capable of reducing a \( n \) dimensional vector to a scalar.

\(^4\)We omit subscripts for clarity.
• Sum (SM) Function is a non-parameterized function that sums the entire vector, returning a scalar as an output.
\[ F(x) = \sum_{i=1}^{n} x_i, \quad \forall x_i \in x \] (10)

• Neural Network (NN) is a fully-connected layer that converts each \( n \) dimensional feature vector as follows:
\[ F(x) = \text{ReLU}(W_c(x) + b_c). \] (11)
where \( W_c \in \mathbb{R}^{nx1} \) and \( b_c \in \mathbb{R} \) are the parameters of the FC layer.

• Factorization Machines (FM) are general purpose machine learning techniques that accept a real-valued feature vector \( x \in \mathbb{R}^n \) and return a scalar output.
\[ F(x) = w_0 + \sum_{i=1}^{n} w_i x_i + \sum_{i=1}^{n} \sum_{j=i+1}^{n} (v_{ij}, v_{ij}) x_i x_j \] (12)
where \( w_0 \in \mathbb{R}, w_i \in \mathbb{R}^n \) and \( \{v_{11}, \ldots, v_{nk}\} \in \mathbb{R}^{nk} \) are the parameters of the FM model. FMs are expressive models that capture pairwise interactions between features using factorized parameters. \( k \) is the number of factors of the FM model. For more details, we refer interested readers to [24].

Note that we do not share parameters across multiple attention casts because each attention cast is aimed at modeling a different view. Our experiments report the above mentioned variants under the model name MCAN (SM), MCAN (NN) and MCAN (FM) respectively.

2.3.3 Multi-Cast. The key idea behind our architecture is the facilitation of \( k \) attention calls (or casts), with each cast augmenting raw word embeddings with a real-valued attentional hint. We formally describe the multi-cast attention mechanism. For each query-document pair, we apply (1) Co-Attention with mean-pooling (2) Co-Attention with max-Pooling and (3) Co-Attention with alignment-pooling. Additionally, we apply Intra-Attention to both query and document individually. Each attention cast produces three scalars (per word) which are concatenated with the word embedding. The final casted feature vector is \( z \in \mathbb{R}^{12} \). As such, for each word \( w_i \), the new word representation becomes \( \hat{w}_i = [w_i; z_i] \).

2.4 Long Short-Term Memory Encoder

Next, the word representations with casted attention \( \hat{w}_1, \hat{w}_2, \ldots, \hat{w}_\ell \) are then passed into a sequential encoder layer. We adopt a standard vanilla long short-term memory (LSTM) encoder:
\[ h_i = \text{LSTM}(u_i, i), \forall i \in \{1, \ldots, \ell\} \] (13)
where \( \ell \) represents the maximum length of the sequence. Notably, the parameters of the LSTM are ‘siamese’ in nature, sharing weights between document and query. The key idea is that the LSTM encoder learns representations that are aware of sequential dependencies by the usage of nonlinear transformations as gating functions. Since LSTMs are standard neural building blocks, we omit technical details in favor of brevity. As such, the key idea behind casting attention as features right before this layer is that it provides the LSTM encoder with hints that provide information such as (1) long-term and global sentence knowledge and (2) knowledge between sentence pairs (document and query).

2.4.1 Pooling Operation. Finally, a pooling function is applied across the hidden states \( \{h_1 \ldots h_\ell\} \) of each sentence, converting the sequence into a fixed dimensional representation.
\[ h = \text{MeanMax}\{h_1 \ldots h_\ell\} \] (14)

We adopt the MeanMax pooling operator, which concatenates the result of the mean pooling and max pooling together. We found this to consistently perform better than using max or mean pooling in isolation.

2.5 Prediction Layer and Optimization

Finally, given a fixed dimensional representation of the document-query pair, we pass their concatenation into a two-layer \( h \)-dimensional highway network. The final prediction layer of our model is computed as follows:
\[ y_{\text{out}} = H_2(H_1([x_q; x_d; x_q \odot x_d; x_q - x_d])) \] (15)
where \( H_1(\cdot), H_2(\cdot) \) are highway network layers with ReLU activation. The output is then passed into a final linear softmax layer.
\[ y_{\text{pred}} = \text{softmax}(W_F \cdot y_{\text{out}} + b_F) \] (16)
where \( W_F \in \mathbb{R}^{nx2} \) and \( b_F \in \mathbb{R}^2 \). The network is then trained using standard multi-class cross entropy loss with L2 regularization.

\[ J(\theta) = -\sum_{i=1}^{N} [y_i \log \hat{y}_i + (1 - y_i) \log(1 - \hat{y}_i)] + \lambda ||\theta||_{L2} \] (17)
where \( \theta \) are the parameters of the network. \( \hat{y} \) is the output of the network. \( ||\theta||_{L2} \) is the L2 regularization and \( \lambda \) is the weight of the regularizer.

3 EMPIRICAL EVALUATION

In our experiments, we aim to answer the following research questions (RQs):

(1) **RQ1** - Does our proposed approach achieve state-of-the-art performance on question answering and conversation modeling tasks? What is the relative improvement over well-established baselines?

(2) **RQ2** - What are the impacts of architectural design on performance? Is the LSTM encoder necessary to make use of the casted features? Does all the variations of co-attention contribute to the overall model performance?

(3) **RQ3** - Can we explain the inner workings of our proposed model? Can we interpret the casted attention features?

3.1 Experiment 1 - Dialogue Prediction

In this first task, we evaluate our model on its ability to successfully predict the next reply in conversations.

3.1.1 Dataset and Evaluation Metric. For this experiment, we utilize the large and well-known large-scale Ubuntu Dialogue Corpus (UDC) [15]. We use the same testing splits provided by Xu et al. [51]. In this task, the goal is to match a sentence with its reply. Following [46], the task mainly utilizes the last two utterances in each conversation, predicting if the latter follows the former. The training set consists of one million message-response pairs with a 1 : 1 positive-negative ratio. The development and testing sets have
3.2.1 Dataset and Evaluation Metric. We utilize the QA dataset from TREC (Text Retrieval Conference). TrecQA is one of the most widely evaluated dataset, competitive and long standing benchmark for QA. This dataset was prepared by Wang et al. [41] and contains 53K QA pairs for training and 1100/1500 pairs for development and testing respectively. Following the recent works, we evaluate on the clean setting as noted by [22]. The evaluation metrics for this task are the MAP (mean average precision) and MRR (mean reciprocal rank) scores which are well-known IR metrics.

3.2.2 Competitive Baselines and Implementation Details. We compare against a large number of competitive baselines, e.g., MLP, DeepMatch [16], ARC-I / ARC-II [9], CNTN [21], MatchPyramid [19], vanilla LSTM, Attentive Pooling LSTM [5], MV-LSTM [37] and finally the state-of-the-art Knowledge Enhanced Hybrid Neural Network (KEHNN) [46]. A detailed description of baselines can be found at [46]. Since testing splits are the same, we report the results directly from [46]. For fair comparison, we set the LSTM encoder size in MCAN to $d = 100$ which makes it equal to the models in [46]. We optimize MCAN with Adam optimizer [13] with an initial learning rate of $3 \times 10^{-4}$. A dropout rate of $0.2$ is applied to all layers except the word embedding layer. The sequences are dynamically truncated or padded to their batch-wise maximums (with a hard limit of 50 tokens). We initialize the word embedding layer with pretrained GloVe embeddings.

3.2.3 Experimental Results. Table 2 reports our results on TrecQA. All MCAN variations outperform all existing state-of-the-art models. Notably, MCAN (FM) is currently the best performing model on this extensively studied dataset. MCAN (NN) comes in second which marginally outperforms the highly competitive and recent IWAN model. Finally, MCAN (SM) remains competitive to IWAN, despite naively summing over casted attention features.

| Model                  | MAP    | MRR    |
|------------------------|--------|--------|
| QA-LSTM (dos Santos et al.) | 0.728  | 0.832  |
| AP-CNN (dos Santos et al.)   | 0.753  | 0.851  |
| LDC Model (Wang et al.)     | 0.771  | 0.845  |
| MPCNN (He et al.)          | 0.777  | 0.836  |
| HyperQA (Tay et al.)       | 0.784  | 0.865  |
| MPCNN + NCE (Rao et al.)   | 0.801  | 0.877  |
| BiMPM (Wang et al.)        | 0.802  | 0.899  |
| IWAN (Shen et al.)         | 0.822  | 0.889  |
| MCAN (SM)                 | 0.827  | 0.880  |
| MCAN (NN)                 | 0.827  | 0.890  |
| MCAN (FM)                 | 0.838  | 0.904  |

Table 2: Performance Comparison on TrecQA (clean) dataset. Best result is in boldface and second best is underlined.

3.3 Experiment 3 - Community Question Answering (cQA)

This task is concerned with ranking answers in community forums. Different from factoid QA, answers are generally subjective instead of factual. Moreover, answer lengths are also much longer.

3.3.1 Dataset and Evaluation. We use the QatarLiving dataset, a well-studied benchmark dataset from SemEval-2016 Task 3 Subtask
A (cQA) and have been used extensively as a benchmark for recent state-of-the-art neural network models for cQA [34, 54]. This is a real world dataset obtained from Qatar Living Forums and comprises 36k training pairs, 2.4k development pairs and 3.6k testing pairs. In this dataset, there are ten answers in each question ‘thread’ which are marked as ‘Good’, ‘Potentially Useful’ or ‘Bad’. Following [54], ‘Good’ is regarded as positive and anything else is regarded as negative labels. We evaluate on two metrics, namely the Precision@1 (P@1) and Mean Average Precision (MAP) metric.

3.3.2 Competitive Baselines and Implementation Details. The key competitors of this dataset are the CNN-based ARC-I/II architecture by Hu et al. [9], the Attentive Pooling CNN [5], Kelp [6] a feature engineering based SVM method, ConvKN [2] a combination of convolutional tree kernels with CNN and finally AI-CNN (Attentive Interactive CNN) [54], a tensor-based attentive pooling neural model. We also compare with the Cross Temporal Recurrent Networks (CTRN) [34], a recently proposed model for ranking QA pairs which have achieved very competitive performance on this dataset. Following [34], we initialize MCAN with domain-specific 200 dimensional word embeddings using the unannotated QatarLiving corpus. Word embeddings are not updated during training. The size of the highway projection layer, LSTM encoder and highway prediction layer are all set to 200. The model is optimized with Adam optimizer with learning rate of $3 \times 10^{-4}$.

| Model          | P@1  | MAP  |
|----------------|------|------|
| ARC-I (Hu et al.) | 0.741 | 0.771 |
| ARC-II (Hu et al.)   | 0.753 | 0.780 |
| AP-CNN (dos Santos et al.) | 0.755 | 0.771 |
| Kelp (Filice et al.)   | 0.751 | 0.792 |
| ConvKN (Barron Cedeno et al.) | 0.755 | 0.777 |
| AI-CNN (Zhang et al.)   | 0.763 | 0.792 |
| CTRN (Tay et al.)     | 0.788 | 0.794 |
| MCAN (SM)            | 0.803 | 0.787 |
| MCAN (NN)            | 0.802 | 0.784 |
| MCAN (FM)            | **0.804** | **0.803** |

Table 3: Performance comparison on QatarLiving dataset for community question answering. Best result is in boldface and second best is underlined.

3.3.3 Experimental Results. Table 3 reports the performance comparison on the QatarLiving dataset. Our best performing MCAN model achieves state-of-the-art performance on this dataset. Performance improvement over recent, competitive neural network baselines is significant. Notably, the improvement of MCAN (FM) over AI-CNN on the P@1 metric is 4.1% and 1.1% in terms of MAP. MCAN (FM) also achieves competitive results relative to the CTRN model. The performance of MCAN (NN) and MCAN (SM) is lower than MCAN (FM) but still remains competitive on this benchmark.

3.4 Experiment 4 - Tweet Reply Prediction

This experiment is concerned with predicting an appropriate reply given a tweet.

3.4.1 Dataset and Evaluation Metrics. We utilize a customer support dataset obtained from Kaggle\(^2\). This dataset contains tweet-reply pairs of tweets to famous brands and their replies. For each Tweet-Reply pair, we randomly selected four tweets as negative samples that originate from the same brand. The dataset is split into 8 : 1 : 1 train-dev-test split. The evaluation metrics for this task are MRR (Mean reciprocal rank) and Precision@1 (accuracy). Unlike previous datasets, there are no published works on this task. As such, we implement the baselines ourselves. We implement standard baselines such as (1) CBOW (sum embeddings) passed into a 2 layer MLP with ReLU activations, (2) standard vanilla LSTM and CNN models and (3) BiLSTM and CNN with standard Co-Attention (AP-BiLSTM, AP-CNN) following [5]. All models minimize the binary cross entropy loss (pointwise) since we found performance to be much better than using ranking loss. We also include the recent AI-CNN (Attentive Interactive CNN) which uses multi-dimensional co-attention. We set all LSTM dimensions to $d = 100$ and the number of CNN filters is 100. The CNN filter width is set to 3. We train all models with Adam optimizer with $3 \times 10^{-4}$ learning rate. Word embeddings are initialized with GloVe and fixed during training. A dropout of 0.2 is applied to all layers except the word embedding layer.

3.4.2 Experimental Results. Table 4 reports our results on the Tweets dataset. MCAN (FM) achieves the top performance by a significant margin. The performance of MCAN (NN) falls short of MCAN (FM), but is still highly competitive. Our best MCAN model outperforms AP-BiLSTM by 3.4% in terms of MRR and 5.3% in terms of P@1. The performance improvement of AI-CNN is even greater, i.e., 8.4% in terms of MRR and 12.5% in terms of P@1. The strongest baseline is AP-BiLSTM which significantly outperforms AI-CNN and AP-CNN.

| Model                  | MRR  | P@1  |
|------------------------|------|------|
| CBOW + MLP             | 0.658 | 0.442 |
| LSTM                   | 0.652 | 0.431 |
| CNN                    | 0.657 | 0.441 |
| AP-CNN (dos Santos et al.) | 0.643 | 0.426 |
| AI-CNN (Zhang et al.)  | 0.675 | 0.465 |
| AP-BiLSTM (dos Santos et al.) | 0.725 | 0.540 |
| MCAN (SM)              | 0.722 | 0.548 |
| MCAN (NN)              | 0.747 | 0.585 |
| MCAN (FM)              | **0.759** | **0.593** |

Table 4: Performance comparison on Reply Prediction on Tweets dataset. Best performance is in boldface and second best is underlined.

3.5 Ablation Analysis

This section aims to demonstrate the relative effectiveness of different components of our proposed MCAN model. Table 5 reports the results on the validation set of the TrecQA dataset. We report the scores of seven different configurations. In (1), we replace all highway layers with regular feed-forward neural networks. In (2),

\(^2\)https://www.kaggle.com/soaxelbrooke/customer-support-on-twitter
we remove the LSTM encoder before the prediction layer. In (3), we remove the entire multi-cast attention mechanism. This is equivalent to removing the twelve attention features. In (4-7), we remove different attention casts, aiming to showcase that removing either one results in some performance drop.

From our ablation analysis, we can easily observe the crucial components to our model. Firstly, we observe that removing MCA entirely significantly decreases the performance (ablation 3). In this case, validation MAP drops from 0.866 to 0.670. As such, our casted attention features contribute a lot to the performance of the model. Secondly, we also observe that the LSTM encoder is necessary. This is intuitive because the goal of MCAN is to provide features as hints for a compositional encoder. As such, removing the LSTM encoder allows our attention hints to go unused. While the upper prediction might still manage to learn from these features, it is still sub-optimal compared to using a LSTM encoder. Thirdly, we observed that removing Max or Mean Co-Attention decreases performance marginally. However, removing the Alignment Co-Attention decreases the performance significantly. As such, it is clear that the alignment-based attention is most important for our model. However, Max, Mean and Intra attention all contribute to the performance of MCAN. Hence, using multiple attention casts can improve performance. Finally, we also note that the highway layers also contribute slightly to performance.

| Setting         | MAP   | MRR  |
|-----------------|-------|------|
| Original        | 0.866 | 0.922|
| (1) Remove Highway | 0.825 | 0.865|
| (2) Remove LSTM  | 0.765 | 0.809|
| (3) Remove MCA   | 0.670 | 0.749|
| (4) Remove Intra | 0.834 | 0.910|
| (5) Remove Align | 0.682 | 0.726|
| (6) Remove Mean  | 0.858 | 0.906|
| (7) Remove Max   | 0.862 | 0.915|

Table 5: Ablation analysis (validation set) on TrecQA dataset.

3.6 In-depth Model Analysis

In this section, we aim to provide insights pertaining to the inner workings of our model. More specifically, we list several observations by visual inspection of the casted attention features. We trained a MCAN model with FM compression and extracted the word-level casted attention features. The features are referred to as $f_i$ where $i \in [1, 12]$. $f_1, f_2, f_3$ are generated from alignment-pooling, $f_4, f_5, f_6$ and $f_7, f_8, f_9$ are generated from max and mean pooled co-attention respectively. $f_{10}, f_{11}, f_{12}$ are generated from intra-attention.

3.6.1 Observation 1: Features learn a Neural Adaptation of Word Overlap. Figure 2 and Figure 3 show a positive and negative QA pair from the TrecQA test set. Firstly, we analyze the first three features $f_1, f_2$ and $f_3$. These features correspond to the alignment attention and multiply, subtract and concat composition respectively. From the figures, we observe that $f_1$ spikes (in the negative direction) when there is a word overlap across sentences, e.g., 'teapot' in 'attack' as in Figure 3 and 'teapot dome scandal' in Figure 2. Hence, $f_1$ (dark blue line) behaves as a neural adaptation of the conventional overlap feature. Moreover, in contrary to traditional binary overlap features, we also notice that the value of the neural word overlap feature is dependent on the word itself, i.e., 'teapot' and 'dome' have different values. As such, it encodes more information over the traditional binary feature.

![Figure 2: Visualization of Casted Attention Features ($f_1, f_2, f_3$) on a positive test sample from TrecQA.](image)

![Figure 3: Visualization of Casted Attention Features ($f_1, f_2, f_3$) on a negative test sample from TrecQA.](image)

3.6.2 Observation 2: Features React to Evidence and Anti-Evidence. While $f_1$ is primarily aimed at modeling overlap, we observe that $f_3$ tries to gather supporting evidence for the given QA pair. In Figure 2, the words 'year' and '1923' have spiked. It also tries to extract key verbs such as 'take place' (question) and 'began' (answer) which are related verbs generally used to describe events. Finally, we observe that $f_2$ (subtractive composition) seems to be searching for anti-evidence, i.e., a contradictory or irrelevant information. However, this appears to be more subtle as compared to $f_1$ and $f_3$. In Figure 3, we note that the words 'died' and 'attack' (answer) have spiked. We find this example particularly interesting because the correct answer '1923' is in fact found in the answer. However, the pair is wrong because the text sample refers to the 'death of Harding' and does not answer the question correctly. In the negative answer, we found that the word 'died' has the highest $f_2$ value. As such, we believe that $f_2$ is actively finding anti-evidence to why this QA pair should be negative. Additionally, irrelevant words such as 'attack' and 'god' experience nudges by $f_2$. Finally, it is good
to note that MCAN classifies these two samples correctly while a standard Bidirectional LSTM does not.

3.6.3 Observation 3: Diversity of Multiple Casts. One of the key motivators for a multi-casted attention is that each attention cast produces features from different views of the sentence pair. While we have shown in our ablation study that all attention casts contributed to the overall performance, this section qualitatively analyzes the output features. Figure 4 shows the casted attention features (answer text) for max-pooled attention \( f_4, f_5, f_6 \) and mean-pooled\(^4\) attention \( f_7, f_8, f_9 \). Note that the corresponding question is the same as Figure 2 and Figure 3 which allows a direct comparison with the alignment-based attention. We observe that both attention casts produce extremely diverse features. More specifically, not only the spikes are all at different words but the overall sequential pattern is very different. We also note that the feature patterns differ a lot from alignment-based attention (Figure 2). While we were aiming to capture more diverse patterns, we also acknowledge that these features are much less interpretable than \( f_1, f_2, f_3 \). Even so, some patterns can still be interpreted, e.g., the value of \( f_5 \) is high for important words and low (negative) whenever the words are generic and unimportant such as ‘to’, ‘the’, ‘a’. Nevertheless, the main objective here is to ensure that these features are not learning identical patterns.

\(^4\)The values on \( f_7 \) are not constant. They appear to be since the max-min range is much smaller than \( f_8 \) and \( f_9 \).

4 RELATED WORK

Learning to rank short document pairs is a long standing problem in IR research. The dominant state-of-the-art models for learning-to-rank today are mostly neural network based models. Neural network models, such as convolutional neural networks (CNN) [9, 29, 32, 40], recurrent neural networks (RNN) [18, 28, 46] or recursive neural networks [38] are used for learning document representations. A parameterized function such as multi-layered perceptrons [27], tensor layers [21] or holographic layers [32] then learns a similarity score between document pairs.

Recent advances in neural ranking models go beyond independent representation learning. There are several main architectural paradigms that invoke interactions between document pairs which intuitively improve performance due to matching at a deeper and finer granularity. The first can be thought of as extracting features from a constructed word-by-word similarity matrix [19, 37]. The second invokes matching across multiple views and perspectives [7, 23, 44]. The third method involves learning pairwise attention weights (i.e., co-attention). In these models, the similarity matrix is used to learn attention weights, learning to attend to each document based on its partner. Attentive Pooling Networks [5] and Attentive Interactive Networks [54] are models that are grounded in this paradigm, utilizing extractive max-pooling to learn the relative importance of a word based on its maximum importance to all words in the other document. The Compare-Aggregate model [42]
used a co-attention model for matching and then a convolutional feature extractor for aggregating features. Notably, other related problem domains such as machine comprehension [26, 43, 50] and review-based recommendation [36] also extensively make use of co-attention mechanisms.

Learning sequence alignments via attention have been also popularized by models in related problem domains such as natural language inference [3, 20, 33]. Notably, MCAN can be viewed as an extension of the CAFE model proposed in [33] for natural language inference. However, the key differences of this work is that (1) the propagated features in MCAN are multi-casted (e.g., multiple co-attention variants are used consecutively) and (2) MCAN is extensively evaluated on a different and diverse set of problem domains and tasks.

There are several other notable and novel classes of model architectures which have been proposed for learning to rank. Examples include knowledge-enhanced models [46, 48], lexical decomposition [45], fused temporal gates [34] and coupled LSTMs [14]. Novel metric learning techniques such as hyperbolic spaces have also been proposed [35], [53] proposed a quantum-like model for matching QA pairs.

Our work is also closely related to the problem domain of ranking for web search, in which a myriad of neural ranking models were also proposed [4, 10, 12, 17, 29, 30, 49]. Ranking models for multi-turn response selection on Ubuntu corpus was also proposed in [47].

### 5 CONCLUSION

We proposed a new state-of-the-art neural model for a myriad of retrieval and matching tasks in the domain of question answering and conversation modeling. Our proposed model is based on a re-imagination of the standard and widely applied neural attention. For the first time, we utilize attention not as a pooling operator but as a form of feature augmentation. We propose three methods to compress attentional matrices into scalar features. Via visualisation and qualitative analysis, we show that these casted features can be interpreted and understood. Our proposed model achieves highly competitive results on four benchmark tasks and datasets. The achievements of our proposed model are as follows: (1) our model obtains the highest performing result on the well-studied TrecQA dataset, (2) our model achieves 9% improvement on Ubuntu dialogue corpus relative to the best existing model, and (3) our model achieves strong results on Community Question Answering and Tweet Reply Prediction.

### REFERENCES

[1] Dimitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. arXiv preprint arXiv:1409.0473 (2014).

[2] Alberto Barrión-Cedeño, Giovanni Da San Martino, Shaqiq R. Joty, Alessandro Moschitti, Fahad Al-Obaidi, Salvatore Romeo, Kateryna Tymoshenko, and Antonio Uva. 2016. ConvKN at SemEval-2016 Task 3: Answer and Question Selection for Question Answering on Arabic and English Fora. In Proceedings of the 10th Conference on Empirical Methods in Natural Language Processing, EMNLP 2016.

[3] Qian Chen, Xiaodan Zhu, Zhen-Hua Ling, Si Wei, Hui Jiang, and Diana Inkpen. 2017. Enhanced LSTM for Natural Language Inference. In Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017.

[4] Mostafa Dehghani, Hamed Zamani, Aliaksei Severyn, Jaap Kamps, and W Bruce Croft. 2017. Neural ranking models with weak supervision. In Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval.

[5] Cicero Nogueira dos Santos, Ming Tan, Bing Xiang, and Bowen Zhou. 2016. Attention Pooling Networks. CoRR abs/1602.03609 (2016).

[6] Simone Filice, Danilo Croce, Alessandro Moschitti, and Roberto Basili. 2016. ReLP at SemEval-2016 Task 3: Learning Semantic Relations between Questions and Answers. In Proceedings of the 10th International Workshop on Semantic Evaluation, SemEval@NAACL-HLT 2016.

[7] Hua He, Kevin Gimpel, and Jimmy J. Lin. 2015. Multi-Perspective Sentence Similarity Modeling with Convolutional Neural Networks. In Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP 2015.

[8] Hua He and Jimmy J. Lin. 2016. Pairwise Word Interaction Modeling with Deep Neural Networks for Semantic Similarity Measurement. In NAACL HLT 2016, The 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, San Diego California, USA, June 12-17, 2016. 937–948.

[9] Baotian Hu, Zhengdong Lu, Hang Li, and Qingcai Chen. 2014. Convolutional Neural Network Architectures for Matching Natural Language Sentences. In Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014.

[10] Po-Sen Huang, Xiaodong He, Jianfeng Gao, Li Deng, Alex Acero, and Larry Heck. 2013. Learning deep structured semantic models for web search using clickthrough data. In Proceedings of the 22nd ACM international conference on Conference on information & knowledge management.

[11] Hai Kui, Andrew Yates, Klaus Berberich, and Gerard de Melo. 2017. PACBR: A Position-Aware Neural IR Model for Relevance Matching. In Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing. 1049–1058.

[12] Hai Kui, Andrew Yates, Klaus Berberich, and Gerard de Melo. 2017. RE-PACBR: A Context and Density-Aware Neural Information Retrieval Model. arXiv preprint arXiv:1706.10192 (2017).

[13] Diederik P. Kingma and Jimmy Ba. 2014. Adam: A Method for Stochastic Optimization. CoRR abs/1412.6980 (2014).

[14] Pengfei Liu, Xipeng Qiu, Yaqian Zhou, Jifan Chen, and Xuanjing Huang. 2016. Modelling Interaction of Sentence Pair with Coupled-LSTMs. In Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, Austin, Texas, USA, November 1-4, 2016. 1703–1712.

[15] Ryan Lowe, Nisan Pow, Julian Serban, and Joelle Pineau. 2015. The ubuntu dialogue corpus: A large dataset for research in unstructured multi-turn dialogue systems. arXiv preprint arXiv:1506.08909 (2015).

[16] Zhengdong Lu and Hang Li. 2013. A deep architecture for matching short texts. In Advances in Neural Information Processing Systems.

[17] Bhaskar Mitra, Fernando Diaz, and Nick Craswell. 2017. Learning to match using local and distributed representations of text for web search. In Proceedings of the 26th International Conference on World Wide Web.

[18] Jonas Mueller and Aditya Thayagarajan. 2016. Siamese Recurrent Architectures for Learning Sentence Similarity. In Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, 2016.

[19] Liang Pang, Yanyan Lan, Jiafeng Gao, Jun Xu, Shexiang Wan, and Xueqi Cheng. 2016. Text Matching as Image Recognition.

[20] Ankur P. Parikh, Oscar Tackstrom, Dipanjan Das, and Jakob Uszkoreit. 2016. A Decomposable Attention Model for Natural Language Inference. In Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, Austin, Texas, USA, November 1-4, 2016.

[21] Xipeng Qiu and Xuanjing Huang. 2015. Convolutional Neural Tensor Network for Community-Based Question Answering. In Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence, IJCAI 2015.

[22] Jinfeng Rao, Hua He, and Jimmy J. Lin. 2016. Noise-Contrastive Estimation for Answer Selection with Deep Neural Networks. In Proceedings of the 25th ACM International on Conference on Information and Knowledge Management, CIKM 2016.

[23] Jinfeng Rao, Wei Yang, Yuah Zhang, Ferhan Ture, and Jimmy Lin. 2018. Multi-Perspective Relevance Matching with Hierarchical ConvNets for Social Media Search. (2018). arXiv:arXiv:1805.08159.

[24] Steffen Rendle. 2010. Factorization machines. In Data Mining (ICDM), 2010 IEEE 10th International Conference on. IEEE, 995–1000.

[25] Tim Rocktäschel, Edward Grefenstette, Karl Moritz Hermann, Tomáš Kočiský, and Phil Blunsom. 2015. Reasoning about entailment with neural attention. arXiv preprint arXiv:1509.06664 (2015).

[26] Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. 2016. Bidirectional attention flow for machine comprehension. arXiv preprint arXiv:1611.01603 (2016).

[27] Aliaksei Severyn and Alessandro Moschitti. 2015. Learning to Rank Short Text Pairs with Convolutional Deep Neural Networks. In Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval, 2015.

[28] Gehua Shen, Yunlin Yang, and Zhi-Hong Deng. 2017. Inter-Weighted Alignment Network for Sentence Pair Modeling. In Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, EMNLP 2017.
