# A Compare-Aggregate Model with Latent Clustering for Answer Selection

Seunghyun Yoon[*]
mysmilesh@snu.ac.kr
Seoul National University
Seoul, Korea

Franck Dernoncourt
franck.dernoncourt@adobe.com
Adobe Research
San Jose, CA, USA

Doo Soon Kim
dkim@adobe.com
Adobe Research
San Jose, CA, USA

Trung Bui
bui@adobe.com
Adobe Research
San Jose, CA, USA

Kyomin Jung
kjung@snu.ac.kr
Seoul National University
Seoul, Korea

## ABSTRACT

In this paper, we propose a novel method for a sentence-level answer-selection task that is a fundamental problem in natural language processing. First, we explore the effect of additional information by adopting a pretrained language model to compute the vector representation of the input text and by applying transfer learning from a large-scale corpus. Second, we enhance the compare-aggregate model by proposing a novel latent clustering method to compute additional information within the target corpus and by changing the objective function from listwise to pointwise. To evaluate the performance of the proposed approaches, experiments are performed with the WikiQA and TREC-QA datasets. The empirical results demonstrate the superiority of our proposed approach, which achieve state-of-the-art performance for both datasets.

## CCS CONCEPTS

• **Information systems → Question answering**; • **Computing methodologies → Natural language processing**.

## KEYWORDS

question answering; natural language processing; information retrieval; deep learning

[*]Work conducted while the author was an intern at Adobe Research.

## 1 INTRODUCTION

Automatic question answering (QA) is a primary objective of artificial intelligence. Recently, research on this task has taken two major directions based on the answer span considered by the model. The first direction (i.e., the fine-grained approach) finds an exact answer to a question within a given passage [7]. The second direction (i.e., the coarse-level approach) is an information retrieval (IR)-based approach that provides the most relevant sentence from a given document in response to a question. In this study, we are interested in building a model that computes a matching score between two text inputs. In particular, our model is designed to undertake an answer-selection task that chooses the sentence that is most relevant to the question from a list of answer candidates. This task has been extensively investigated by researchers because it is a fundamental task that can be applied to other QA-related tasks [1, 5, 9, 11, 12, 15].

However, most previous answer-selection studies have employed small datasets [14, 17] compared with the large datasets employed for other natural language processing (NLP) tasks [4, 7]. Therefore, the exploration of sophisticated deep learning models for this task is difficult.

To fill this gap, we conduct an intensive investigation with the following directions to obtain the best performance in the answer-selection task. First, we explore the effect of additional information by adopting a pretrained language model (**LM**) to compute the vector representation of the input text. Recent studies have shown that replacing the word-embedding layer with a pretrained language model helps the model capture the contextual meaning of words in the sentence [2, 6]. Following this study, we select an ELMo [6] language model for this study. We investigate the applicability of transfer learning (**TL**) using a large-scale corpus that is created for a relevant-sentence-selection task (i.e., question-answering NLI (QNLI) dataset [13]). Second, we further enhance one of the baseline models, **Comp-Clip** [1] (refer to the discussion in 3.1), for the target QA task by proposing a novel latent clustering (**LC**) method. The **LC** method computes latent cluster information for target samples by creating a latent memory space and calculating the similarity between the sample and the memory. By an end-to-end learning process with the answer-selection task, the **LC** method assigns *true*-label question-answer pairs to similar clusters. In this manner, a model will have further information for matching sentence pairs, which increases the total model performance. Last,
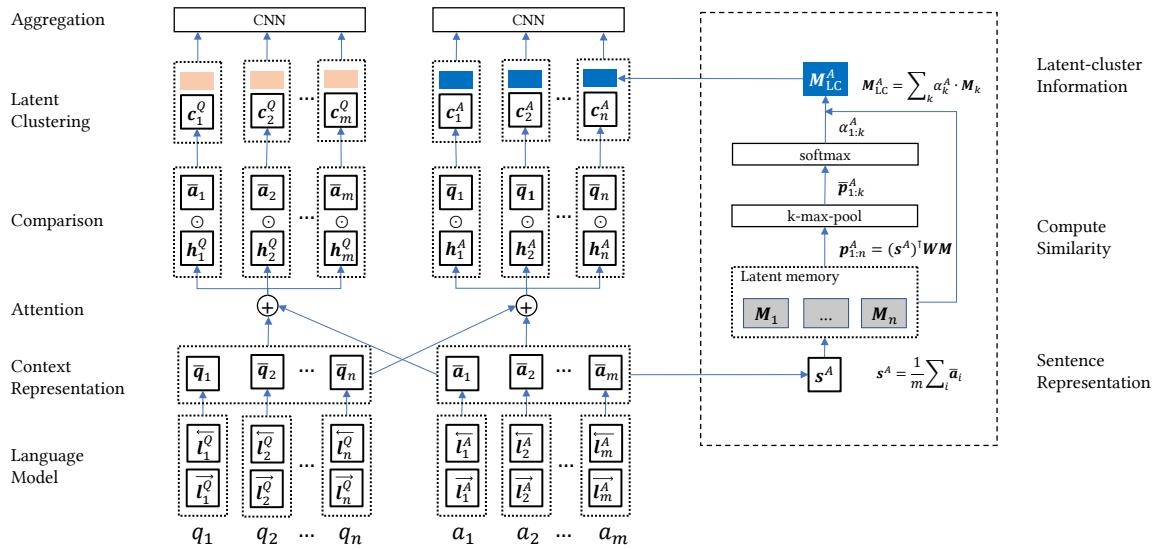
**Figure 1: The architecture of the model. The dotted box on the right shows the process through which the latent-cluster information is computed and added to the answer. This process is also performed in the question part but is omitted in the figure. The latent memory is shared in both processes.**

we explore the effect of different objective functions (listwise and pointwise learning). In contrast to previous research [1], we observe that the pointwise learning approach performs better than the listwise learning approach when we apply our proposed methods. Extensive experiments are conducted to investigate the efficacy and properties of the proposed methods and show the superiority of our proposed approaches for achieving state-of-the-art performance with the WikiQA and TREC-QA datasets.

## 2 RELATED WORK

Researchers have investigated models based on neural networks for question-answering tasks. One study employs a Siamese architecture that utilizes an encoder (e.g., RNN or CNN) to compute vector representations of the question and the answer. The affinity score is calculated based on these vector representations [4]. To improve the model performance by enabling the use of information from one sentence (e.g., a question or an answer) in computing the representation of another sentence, researchers included the attention mechanism in their models [8, 10, 16].

Another line of research includes the compare-aggregate framework [15]. In this framework, first, vector representations of each sentence are computed. Second, these representations are compared. Last, the results are aggregated to calculate the matching score between the question and the answer [1, 9, 12].

In this study, unlike the previous research, we employ a pretrained language model and a latent-cluster method to help the model understand the information in the question and the answer.

## 3 METHODS

### 3.1 Comp-Clip Model

In this paper, we are interested in estimating the matching score $f(y|\mathbf{Q}, \mathbf{A})$, where $y$, $\mathbf{Q} = \{q_1, ..., q_n\}$ and $\mathbf{A} = \{a_1, ..., a_m\}$ represent

the label, the question and the answer, respectively. We select the model from [1], which is referred to as the **Comp-Clip** model, as our baseline model. The model consists of the following four parts:

**Context representation:** The question $\mathbf{Q} \in \mathbb{R}^{d \times Q}$ and answer $\mathbf{A} \in \mathbb{R}^{d \times A}$, (where $d$ is a dimensionality of word embedding and Q and A are the length of the sequence in $\mathbf{Q}$ and $\mathbf{A}$, respectively), are processed to capture the contextual information and the word as follows:

$$
\begin{aligned}
\overline{\mathbf{Q}} &= \sigma(\mathbf{W}^i \mathbf{Q}) \odot \tanh(\mathbf{W}^u \mathbf{Q}), \\
\overline{\mathbf{A}} &= \sigma(\mathbf{W}^i \mathbf{A}) \odot \tanh(\mathbf{W}^u \mathbf{A}),
\end{aligned}
\tag{1}
$$

where $\odot$ denotes element-wise multiplication, and $\sigma$ is the sigmoid function. The $\mathbf{W} \in \mathbb{R}^{l \times d}$ is the learned model parameter.

**Attention:** The soft alignment of each element in $\overline{\mathbf{Q}} \in \mathbb{R}^{l \times Q}$ and $\overline{\mathbf{A}} \in \mathbb{R}^{l \times A}$ are calculated using dynamic-clip attention [1]. We obtain the corresponding vectors $\mathbf{H}^Q \in \mathbb{R}^{l \times A}$ and $\mathbf{H}^A \in \mathbb{R}^{l \times Q}$.

$$
\begin{aligned}
\mathbf{H}^Q &= \overline{\mathbf{Q}} \cdot \text{softmax}((\mathbf{W}^q \overline{\mathbf{Q}})^\top \overline{\mathbf{A}}), \\
\mathbf{H}^A &= \overline{\mathbf{A}} \cdot \text{softmax}((\mathbf{W}^a \overline{\mathbf{A}})^\top \overline{\mathbf{Q}}).
\end{aligned}
\tag{2}
$$

**Comparison:** A comparison function is used to match each word in the question and answer to a corresponding attention-applied vector representation:

$$
\begin{aligned}
\mathbf{C}^Q &= \overline{\mathbf{A}} \odot \mathbf{H}^Q, \ (\mathbf{C}^Q \in \mathbb{R}^{l \times A}), \\
\mathbf{C}^A &= \overline{\mathbf{Q}} \odot \mathbf{H}^A, \ (\mathbf{C}^A \in \mathbb{R}^{l \times Q}),
\end{aligned}
\tag{3}
$$

where $\odot$ denotes element-wise multiplication.

**Aggregation:** We aggregate the vectors from the comparison layer using CNN [3] with $n$-types of filters and calculate the matching

score between $\mathbf{Q}$ and $\mathbf{A}$.

$$\mathbf{R}^Q = \text{CNN}(\mathbf{C}^Q), \ \mathbf{R}^A = \text{CNN}(\mathbf{C}^A),$$
$$\text{score} = \sigma([\mathbf{R}^Q; \mathbf{R}^A]^\intercal \mathbf{W}), \tag{4}$$

where [;] denotes concatenation of each vector $\mathbf{R}^Q \in \mathbb{R}^{nl}$ and $\mathbf{R}^A \in \mathbb{R}^{nl}$. The $\mathbf{W} \in \mathbb{R}^{2nl \times 1}$ is the learned model parameter.

## 3.2 Proposed Approaches

To achieve the best performance in the answer-selection task, we propose four approaches: adding a pretrained **LM**; adding the **LC** information of each sentence as auxiliary knowledge; applying **TL** to benefit from large-scale data; and modifying the objective function from listwise to pointwise learning. Figure 1 depicts the total architecture of the proposed model.

**Pretrained Language Model (LM):** Recent studies have shown that replacing the word embedding layer with a pretrained **LM** helps the model capture the contextual meaning of the words in the sentence [2, 6]. We select an ELMo [6] language model and replace the previous word embedding layer with the ELMo model as follows: $\mathbf{L}^Q = \text{ELMo}(\mathbf{Q})$, $\mathbf{L}^A = \text{ELMo}(\mathbf{A})$. These new representations—$\mathbf{L}^Q$ and $\mathbf{L}^A$—are substituted for $\mathbf{Q}$ and $\mathbf{A}$, respectively, in equation (1).

**Latent Clustering (LC) Method:** We assume that extracting the **LC** information of the text and using it as auxiliary information will help the neural network model analyze the corpus. The dotted box in figure 1 shows the proposed **LC** method. We create $n$-many latent memory vectors $\mathbf{M}_{1:n}$ and calculate the similarity between the sentence representation and each latent memory vector. The latent-cluster information of the sentence representation will be obtained using a weighted sum of the latent memory vectors according to the calculated similarity as follows:

$$\begin{aligned}
\mathbf{p}_{1:n} &= \mathbf{s}^\intercal \mathbf{W} \mathbf{M}_{1:n}, \\
\overline{\mathbf{p}}_{1:k} &= k\text{-max-pool}(\mathbf{p}_{1:n}), \\
\alpha_{1:k} &= \text{softmax}(\overline{\mathbf{p}}_{1:k}), \\
\mathbf{M}_{\text{LC}} &= \Sigma_k \overline{\alpha}_k \mathbf{M}_k,
\end{aligned} \tag{5}$$

where $\mathbf{s} \in \mathbb{R}^d$ is a sentence representation, $\mathbf{M}_{1:n} \in \mathbb{R}^{d' \times n}$ indicates the latent memory, and $\mathbf{W} \in \mathbb{R}^{d \times d'}$ is the learned model parameter.

We apply the **LC** method and extract cluster information from each question and answer. This additional information is added to each of the final representations in the comparison part (see 3.1) as follows:

$$\begin{aligned}
\mathbf{M}_{\text{LC}}^Q &= f((\Sigma_i \overline{q}_i)/n), \ \overline{q}_i \subset \overline{\mathbf{Q}}_{1:n}, \\
\mathbf{M}_{\text{LC}}^A &= f((\Sigma_i \overline{a}_i)/m), \ \overline{a}_i \subset \overline{\mathbf{A}}_{1:m}, \\
\mathbf{C}_{\text{new}}^Q &= [\mathbf{C}^Q; \mathbf{M}_{\text{LC}}^Q], \ \mathbf{C}_{\text{new}}^A = [\mathbf{C}^A; \mathbf{M}_{\text{LC}}^A],
\end{aligned} \tag{6}$$

where $f$ is the **LC** method (in equation 5) and [;] denotes the concatenation of each vector. These new representations—$\mathbf{C}_{\text{new}}^Q$ and $\mathbf{C}_{\text{new}}^A$—are substituted for $\mathbf{C}^Q$ and $\mathbf{C}^A$ in equation (4). Note that we average word-embedding to obtain sentence representation in the previous equation.

**Transfer Learning (TL):** To observe the efficacy in a large dataset, we apply transfer learning using the question-answering NLI (QNLI) corpus [13]. We train the **CompClip** model with the QNLI corpus

**Table 1: Properties of the dataset.**

| Dataset | Listwise pairs | | | Pointwise pairs | | |
|---------|------|-----|------|-------|------|------|
| | train | dev | test | train | dev | test |
| WikiQA | 873 | 126 | 243 | 8.6k | 1.1k | 2.3k |
| TREC-QA | 1.2k | 65 | 68 | 53k | 1.1k | 1.4k |
| QNLI | 86k | 10k | - | 428k | 169k | - |

and then fine-tune the model with target corpora, such as the Wik-iQA [17] and TREC-QA [14] datasets.

**Pointwise Learning to Rank:** Previous research adopts a listwise learning approach. With a dataset that consists of a question, $\mathbf{Q}$, a related answer set, $\mathbf{A} = \{\mathbf{A}_1, ..., \mathbf{A}_N\}$, and a target label, $\mathbf{y} = \{y_1, ..., y_N\}$, a matching score is computed using equation (4). This approach applies KL-divergence loss to train the model as follows:

$$\begin{aligned}
\text{score}_i &= \text{model}(\mathbf{Q}, \mathbf{A}_i), \\
\mathbf{S} &= \text{softmax}([\text{score}_1, ..., \text{score}_i]), \\
\text{loss} &= \Sigma_{n=1}^N \text{KL}(\mathbf{S}_n || \mathbf{y}_n),
\end{aligned} \tag{7}$$

where $i$ is the number of answer candidates for the given question and $N$ is the total number of samples employed during training.

In contrast, we pair each answer candidate to the question and compute the cross-entropy loss to train the model as follows:

$$\text{loss} = -\Sigma_{n=1}^N y_n \log(\text{score}_n), \tag{8}$$

where $N$ is the total number of samples used during training. Using this approach, the number of training instances for a single iteration increases, as shown in table 1.

## 4 EMPIRICAL RESULTS

We regard all tasks as relevant answer selections for the given questions. Following the previous study, we report the model performance as the mean average precision (MAP) and the mean reciprocal rank (MRR). To test the performance of the model, we utilize the TREC-QA, WikiQA and QNLI datasets [13, 14, 17].

### 4.1 Comparison with Other Methods

Table 2 shows the model performance for the WikiQA and TREC-QA datasets. For the Compare-Aggregate (2016), Comp-Clip (2017), IWAN (2017) and IWAN+sCARNN (2018) models, we measure the performance on the WikiQA dataset using the authors' implementations (marked by * in the table). Unlike previous studies, we report our results for both the dev dataset and the test dataset because we note a performance gap between these datasets. While training the model, we apply an early stop that is based on the performance of the dev dataset and measure the performance on the test dataset. Because **Comp-Clip** [1] is our baseline model, we implement it from scratch and achieve a performance that is similar to that of the original paper.

**WikiQA:** For the WikiQA dataset, the pointwise learning approach shows a better performance than the listwise learning approach. We combine **LM** with the base model (**Comp-Clip** +**LM**) and observe a significant improvement in performance in terms of MAP (0.714

**Table 2: Model performance (the top 3 scores are marked in bold for each task). We evaluate model [1, 9, 12, 15] on the WikiQA corpus using author's implementation (marked by \*). For TREC-QA case, we present reported results in the original papers.**

| Model | WikiQA | | | | TREC-QA | | | |
|---|---|---|---|---|---|---|---|---|
| | MAP | | MRR | | MAP | | MRR | |
| | dev | test | dev | test | dev | test | dev | test |
| Compare-Aggregate (2017) [15] | 0.743* | 0.699* | 0.754* | 0.708* | - | - | - | - |
| Comp-Clip (2017) [1] | 0.732* | 0.718* | 0.738* | 0.732* | - | 0.821 | - | 0.899 |
| IWAN (2017) [9] | 0.738* | 0.692* | 0.749* | 0.705* | - | 0.822 | - | 0.899 |
| IWAN + sCARNN (2018) [12] | 0.719* | 0.716* | 0.729* | 0.722* | - | 0.829 | - | 0.875 |
| MCAN (2018) [11] | - | - | - | - | - | 0.838 | - | **0.904** |
| Question Classification (2018) [5] | - | - | - | - | - | **0.865** | - | 0.904 |
| **Listwise Learning to Rank** | | | | | | | | |
| Comp-Clip (our implementation) | 0.756 | 0.708 | 0.766 | 0.725 | 0.750 | 0.744 | 0.805 | 0.791 |
| Comp-Clip (our implementation) + LM | 0.783 | 0.748 | 0.791 | 0.768 | 0.825 | 0.823 | 0.870 | 0.868 |
| Comp-Clip (our implementation) + LM + LC | 0.787 | 0.759 | 0.793 | 0.772 | 0.841 | 0.832 | 0.877 | 0.880 |
| Comp-Clip (our implementation) + LM + LC +TL | 0.822 | **0.830** | 0.836 | **0.841** | 0.866 | 0.848 | 0.911 | 0.902 |
| **Pointwise Learning to Rank** | | | | | | | | |
| Comp-Clip (our implementation) | 0.776 | 0.714 | 0.784 | 0.732 | 0.866 | 0.835 | 0.933 | 0.877 |
| Comp-Clip (our implementation) + LM | 0.785 | 0.746 | 0.789 | 0.762 | 0.872 | 0.850 | 0.930 | 0.898 |
| Comp-Clip (our implementation) + LM + LC | 0.782 | **0.764** | 0.785 | **0.784** | 0.879 | **0.868** | 0.942 | **0.928** |
| Comp-Clip (our implementation) + LM + LC +TL | 0.842 | **0.834** | 0.845 | **0.848** | 0.913 | **0.875** | 0.977 | **0.940** |

to 0.746 absolute). When we add the **LC** method (**Comp-Clip** +**LM** +**LC**), the best previous results are surpassed in terms of MAP (0.718 to 0.764 absolute). We achieve a vast improvement in performance in terms of the MAP (0.764 to 0.834 absolute) by including the **TL** approach (**Comp-Clip** + **LM** + **LC** + **TL**).

**TREC-QA:** The pointwise learning approach also shows excellent performance with the TREC-QA dataset. As shown in table 1, the TREC-QA dataset has a larger number of answer candidates per question. We assume that this characteristic prevents the model from handling the dataset with a listwise learning approach. As in the WikiQA case, we achieve additional performance gains in terms of the MAP as we apply **LM**, **LC**, and **TL** (0.850, 0.868 and 0.875, respectively). In particular, our model outperforms the best previous result when we add **LC** method, (**Comp-Clip** +**LM** +**LC**) in terms of MAP (0.865 to 0.868).

## 5  CONCLUSION

In this study, our proposed method achieves state-of-the-art performance for both the WikiQA dataset and TREC-QA dataset. We show that leveraging a large amount of data is crucial for capturing the contextual representation of input text. In addition, we show that the proposed latent clustering method with a pointwise objective function significantly improves the model performance in the sentence-level QA task.

## ACKNOWLEDGMENTS

## REFERENCES

[1] Weijie Bian, Si Li, Zhao Yang, Guang Chen, and Zhiqing Lin. 2017. A compare-aggregate model with dynamic-clip attention for answer selection. In *CIKM*. ACM, 1987–1990.

[2] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *NAACL*. 4171–4186.

[3] Yoon Kim. 2014. Convolutional Neural Networks for Sentence Classification. In *EMNLP*. 1746–1751.

[4] Ryan Lowe, Nissan Pow, Iulian Serban, and Joelle Pineau. 2015. The Ubuntu Dialogue Corpus: A Large Dataset for Research in Unstructured Multi-Turn Dialogue Systems. In *SIGDIAL*. 285–294.

[5] Harish Tayyar Madabushi, Mark Lee, and John Barnden. 2018. Integrating Question Classification and Deep Learning for improved Answer Selection. In *ACL*. 3283–3294.

[6] Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep Contextualized Word Representations. In *NAACL*. 2227–2237.

[7] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ Questions for Machine Comprehension of Text. In *EMNLP*. 2383–2392.

[8] Cicero dos Santos, Ming Tan, Bing Xiang, and Bowen Zhou. 2016. Attentive pooling networks. *arXiv preprint arXiv:1602.03609* (2016).

[9] Gehui Shen, Yunlun Yang, and Zhi-Hong Deng. 2017. Inter-weighted alignment network for sentence pair modeling. In *EMNLP*. 1179–1189.

[10] Ming Tan, Cicero dos Santos, Bing Xiang, and Bowen Zhou. 2015. Lstm-based deep learning models for non-factoid answer selection. *arXiv preprint arXiv:1511.04108* (2015).

[11] Yi Tay, Luu Anh Tuan, and Siu Cheung Hui. 2018. Multi-Cast Attention Networks. In *SIGKDD*. ACM, 2299–2308.

[12] Quan Hung Tran, Tuan Lai, Gholamreza Haffari, Ingrid Zukerman, Trung Bui, and Hung Bui. 2018. The Context-dependent Additive Recurrent Neural Net. In *NAACL*, Vol. 1. 1274–1283.

[13] Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R Bowman. 2018. GLUE: A Multi-Task Benchmark and Analysis Platform for Natural Language Understanding. In *EMNLP*. 353.

[14] Mengqiu Wang, Noah A Smith, and Teruko Mitamura. 2007. What is the Jeopardy model? A quasi-synchronous grammar for QA. In *EMNLP-CoNLL*.

[15] Shuohang Wang and Jing Jiang. 2017. A compare-aggregate model for matching text sequences. In *ICLR*.

[16] Zhiguo Wang, Wael Hamza, and Radu Florian. 2017. Bilateral multi-perspective matching for natural language sentences. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*. AAAI Press, 4144–4150.

[17] Yi Yang, Wen-tau Yih, and Christopher Meek. 2015. Wikiqa: A challenge dataset for open-domain question answering. In *EMNLP*. 2013–2018.