Semantic matching in machine reading comprehension: An empirical study

Qian Liu a,∗, Rui Mao a, Xiubo Geng b, Erik Cambria a

a School of Computer Science and Engineering, Nanyang Technological University, Singapore
b STCA NLP Group, Microsoft Company, China

ARTICLE INFO

Keywords:
Natural language processing
Machine reading comprehension
Question answering
Semantic matching

ABSTRACT

Machine reading comprehension (MRC) is a challenging task in the field of artificial intelligence. Most existing MRC works contain a semantic matching module, either explicitly or intrinsically, to determine whether a piece of context answers a question. However, there is scant work which systematically evaluates different paradigms using semantic matching in MRC. In this paper, we conduct a systematic empirical study on semantic matching. We formulate a two-stage framework which consists of a semantic matching model and a reading model, based on pre-trained language models. We compare and analyze the effectiveness and efficiency of using semantic matching modules with different setups on four types of MRC datasets. We verify that using semantic matching before a reading model improves both the effectiveness and efficiency of MRC. Compared with answering questions by extracting information from concise context, we observe that semantic matching yields more improvements for answering questions with noisy and adversarial context. Matching coarse-grained context to questions, e.g., paragraphs, is more effective than matching fine-grained context, e.g., sentences and spans. We also find that semantic matching is helpful for answering who/where/when/what/how/which questions, whereas it decreases the MRC performance on why questions. This may imply that semantic matching helps to answer a question whose necessary information can be retrieved from a single sentence. The above observations demonstrate the advantages and disadvantages of using semantic matching in different scenarios.

1. Introduction

Machine reading comprehension (MRC) is a task that aims at building a system to read text and understand the meaning, then automatically answers questions posed by humans (Hermann, et al., 2015; Li, Li, Pandelea, Ge, Zhu, & Cambria, 2023; Wen, Jiang, Tu, Liu, & Cambria, 2023). Unlike traditional search engines that return a list of relevant snippets or hyperlinks, an MRC system provides a precise answer to a user’s question. It offers better user experiences and information retrieval efficiency in many practical applications. In addition to its utility in practice, MRC potentially drives the development of natural language understanding, such as coreference resolution (Wu, Wang, Yuan, Wu, & Li, 2020), dependency parsing (Gan, et al., 2022), and dialogue system (Li, Li, Chen, & Lin, 2020), and opinion mining (Zhou, Liao, Gao, Jie, & Lu, 2021). The inputs of MRC systems are usually questions from a wide range of domains. The evidence to answer questions exists in a large number of different types of corpora, such as Wikipedia documents, news, and web pages. As shown in Fig. 1, given a question, a typical MRC system first retrieves context from corpora, then a reading model is trained to infer the answer based on the retrieved context.

∗ Corresponding author.
E-mail addresses: liu.qian@ntu.edu.sg (Q. Liu), rui.mao@ntu.edu.sg (R. Mao), xigeng@microsoft.com (X. Geng), cambria@ntu.edu.sg (E. Cambria).

https://doi.org/10.1016/j.ipm.2022.103145
Received 24 May 2022; Received in revised form 29 August 2022; Accepted 30 October 2022
Available online 18 November 2022
0306-4573/© 2022 Elsevier Ltd. All rights reserved.
Though semantic matching has been widely used in MRC, a little works systematically evaluated different semantic matching application paradigms, e.g., the granularity of semantic matching (e.g., paragraph-level, sentence-level, or span-level), and the position of a semantic matching module in an MRC model (before or after a reading module). It is also unclear to what extent a semantic matching module is helpful to different types of questions, e.g., who/where/when/what/how/which/why questions, and different types of source passages for question answering, e.g., concise, noisy, and adversary passages. To bridge this gap, we conduct a systematic empirical study on semantic matching and rethink its significance in MRC. We investigate the following research questions (RQs):

- RQ1: Does semantic matching improve MRC performance in all cases?
• RQ2: Which granularity of semantic matching is more effective?
• RQ3: Should an MRC system conduct semantic matching first or reading first?
• RQ4: What types of passages are more suitable for introducing a semantic matching module?
• RQ5: What types of questions are more suitable for introducing a semantic matching module?

The internal connections between five RQs are as follows. First, we first demonstrate the general applicability of semantic matching in MRC (RQ1), then we contrast various semantic matching paradigms (RQ2 and RQ3), and finally, we investigate the efficacy of semantic matching in different types of MRC scenarios (RQ4 and RQ5). Given the above RQs, our key findings are demonstrated as follows:

1. Semantic matching improves MRC in most of our experiments. The improvements are especially substantial in the presence of noisy and adversarial context, and relatively weak encoders.

2. Paragraph-level semantic matching is more effective than sentence-level and span-level, since paragraphs provide more contextual information for the inference of a downstream reading module. However, a fine-grained retrieved context on sentence-level or span-level may filter out necessary answer spans in the semantic matching stage.

3. The matching-then-reading framework outperforms the reading-then-matching framework in terms of both efficiency and effectiveness (these two frameworks are defined and detailed in Section 3.1). The reason for the improvement in efficiency is that a large portion of the original input (e.g., about 98% for Open-SQuAD Chen, Fisch, Weston, & Bordes, 2017) is filtered out by a semantic matching model, and thus the reading model only needs to read very few context. Moreover, although a large percentage of input is filtered out by the matching model, almost all the relevant context can be recalled. Therefore, in the matching-then-reading framework, the reading model can concentrate on the most relevant context and yield more accurate answers than the reading-then-matching framework.

4. Some MRC datasets may provide concise passages, e.g., SQuAD-2.0 (Rajpurkar, Jia, & Liang, 2018), where one short passage is provided to answer a question. For this dataset, semantic matching brings negative impacts, because the given concise passages have provide distilled information from large corpora. In contrast, semantic matching is more useful for the datasets that contain noisy and adversary information in their context, e.g., Open-SQuAD (Chen et al., 2017), Natural Questions (Kwiatkowski, et al., 2019), and Adversarial-SQuAD (Jia & Liang, 2017). These information is confusing to answer questions for machines. Thus, semantic matching can improve MRC in these scenarios.

5. Semantic matching is more suitable for who/what/where/when/how and which questions, such as “Who expounded the Three Laws of Motion?”, which can be answered by using the information from a single sentence. Semantic matching hurts the performance of a reading model for answering why questions, such as “Why is breathing oxygen in space craft not dangerous to health?”, because answering why questions usually need to reason from multiple sentences. A semantic matching module is ineffective in retrieving those context with multiple useful sentences.

The rest of this paper is arranged as follows: Section 2 provides an overview of the related works on MRC; Section 3 describes comparable setups; Section 4 illustrates employed datasets and implementations; Section 5 presents the findings; finally, Section 6 provides concluding remarks.

2. Related works

In this section, we briefly introduce the extractive MRC task, after which we present the representative works on employing semantic matching in MRC.

2.1. MRC

MRC has attracted increasing attention over the past few years. It aims to teach machines to understand a text like a human. A typical task is extractive question answering, which is defined as answering a question according to a piece of context (such as a passage or a document), where “extractive” stresses the condition that a question can be answered by a span in the context. Recently, a large number of extractive MRC datasets have been constructed, where the questions and answer spans are annotated by humans. For example, the SQuAD 1.1 dataset (Rajpurkar, Zhang, Lopyrev, & Liang, 2016) consists of questions posed by crowdworkers on a set of Wikipedia articles, where the answer to every question is a span from the corresponding reading passage. The later version, SQuAD 2.0 (Rajpurkar et al., 2018), contains unanswerable questions so systems must not only answer questions when possible, but also determine when no answer is supported by the paragraph and abstain from answering. The Natural Questions (Kwiatkowski, et al., 2019) dataset contains questions from real users. Systems are required to read and comprehend an entire Wikipedia article that may or may not contain the answer to a question. These high-quality datasets spur the development of machine reading comprehension. Consequently, various neural reading comprehension systems have been devised.

Generally, to address the extractive MRC task, a natural language understanding module should capture the interactions between the questions and the passage and gather evidence from useful passages. Early studies have developed a number of deep neural architectures for encoding questions and passages based on RNN (Wang, Yang, Wei, Chang, & Zhou, 2017), BiLSTM (Seo, Kembhavi, Farhadi, & Hajishirzi, 2017) and CNN (Lai, et al., 2019; Zheng, Zheng, et al., 2020). More recently, MRC has stepped into a new stage with the help of pre-trained language models (PLMs), such as ELMo (Peters, et al., 2018), BERT (Devlin, Chang, Lee, & Toutanova, 2019), RoBERTa (Liu, et al., 2019), and XLNet (Yang, et al., 2019). These language models take advantage of deep architectures, training on large scale unlabeled data to capture complex language phenomena (Mao, Lin, & Guerin, 2019), further offering substantial performance boosts on MRC.
The PLM-based methods jointly encode questions and context, followed by task-specific fine-tuning, and achieve state-of-the-art performance in various leaderboards, such as SQuAD1 (Rajpurkar et al., 2016) and Natural Questions2 (Kwiatkowski et al., 2019). Based on PLMs, some works have been developed to improve MRC from different aspects, such as incorporating commonsense knowledge (Sun, Yu, Chen, Yu, & Cardie, 2022), multi-hop reasoning (Min et al., 2019), and employing a semantic matching module. In this work, we focus on revealing the importance of semantic matching in MRC. Previous works mainly take RNN or BiLSTM as the encoder for both reading and matching modules, however, how these approaches perform when coupled with a powerful PLM is unclear. To adapt to the current research stage, we conduct a comprehensive study of semantic matching, especially in relation to its performance when powerful PLMs are used as the encoder.

2.2. Semantic matching in MRC

In NLP, semantic matching is undertaken to identify whether two pieces of text are semantically related, and is widely used in diverse tasks such as text classification (Chen, Ma, Lin, & Yan, 2021; Liu, Lu, et al., 2021), sentiment analysis (Cambria, Liu, Decherchi, Xing, & Kwok, 2022; Chen & Xie, 2020; Chen, Xie, Cheng, & Li, 2022; Chen, Xie, Li, & Cheng, 2021), dialogue systems (Ni, Pandelea, Young, Zhou, & Cambria, 2022; Young, Xing, Pandelea, Ni, & Cambria, 2022) and natural language inference (Bowman, Angeli, Potts, & Manning, 2015). In extractive MRC, various methods have been designed with semantic matching to improve the natural language understanding ability.

First, some methods explicitly employ a semantic matching module to retrieve relevant document content, and then extract answers from the retrieved context using a reading module. We name this type of framework as matching-then-reading. For example, Choi et al. (2017) presented a coarse-to-fine framework that first selects relevant sentences and then uses a recurrent architecture on these sentences to find the answer. Swayamdipta, Parikh, and Kwiatkowski (2018) constructed a cascade of lightweight models to handle longer evidence documents and aggregated information from multiple occurrences of answer spans throughout the documents. Wang, Yu, Guo, et al. (2018) proposed a joint model with a document ranker, trained with reinforcement learning. Zhong, Xiong, Keskar, and Socher (2019) proposed a multi-evidence model which consists of a coarse-grained module to find relevant answers and a fine-grained module to score all candidate answers. Min et al. (2018) showed that questions in most existing datasets can be answered with a small set of sentences. They employed a sentence selector to generate a minimal set of sentences to feed into an answer generator. Nie, Wang, and Bansal (2019) revealed the importance of semantic retrieval at different levels of granularity. They proposed a simple yet effective pipeline system that retrieves coarse-grained passages on paragraph-level first, and then retrieves fine-grained passages on a sentence-level from the retrieved paragraphs in the last step. Garg et al. (2020) designed an answer sentence selection model based on transfer learning. It is pre-trained with a semantic matching task on large-scale MRC datasets and can be adapted to other MRC datasets. Mrini et al. (2021) introduced the Tree Aggregation Transformer to improve the performance of answer sentence selection in MRC.

Several methods employ semantic matching to re-rank the candidate answers generated by a reading model. A reading model first generates candidate answer spans. Then a semantic matching module re-ranks these spans. We define this type of framework as reading-then-matching. For example, Trischler et al. (2016) designed an end-to-end model which generated a small set of candidate answers and formulated hypotheses using the proposed candidates and the question, then reranked the hypotheses based on their estimated concordance with the supporting text. Jia and Liang (2017) showed that previous reading methods are easily confused by similar passages or sentences and the answer verification module is helpful to alleviate this problem. Wang, Liu, et al. (2018) designed a weighted combination of three re-rankers and re-scored the concatenation of all passages with a particular answer, using a sequential model. Hu, Peng, Huang, and Li (2019) proposed a unified question answering model with a span-level semantic matching model to re-rank the candidate answers. Kratzwald et al. (2019) proposed a pipeline question answering method with a re-ranking stage. In the re-ranking stage, they aggregated features from previous information retrieval and machine comprehension modules, ranking based on a two-layer feed-forward network and a modified pair-wise ranking loss. Iyer et al. (2021) designed a re-rank method named RECONSIDER, which is trained on positive and negative examples, extracted from high confidence predictions. They used in-passage span annotations to perform span-focused re-ranking over a smaller candidate set.

Moreover, some works also implicitly employ the semantic matching module in the reading models. For example, Liu, Gong, et al. (2020) showed that, when reading long documents to answer questions, most answers are only related to a few words in one paragraph. Thus, they designed a RikiNet method with a paragraph-level matching module. Zheng, Wen, et al. (2020) leveraged Graph Attention Networks to learn paragraph-level representations and token-level representations to predict the paragraphs that contain the answers to the input questions, and then yield answer spans. Liu, Geng, et al. (2021) designed an MRC model with paragraph-level and sentence-level semantic matching to improve the accuracy of answer extraction. These methods represent a paragraph or a sentence considering their global context, instead of individual question–question pairs. As such, they are more complex but more effective for MRC.

In summary, a large number of previous works have developed different semantic matching paradigms for MRC, e.g., matching-then-reading and reading-then-matching. However, little work has been done to systematically evaluate the effectiveness and efficiency of semantic matching and compare the utilities of semantic matching with different setups, e.g., the different granularity of semantic matching retrieved context; the different positions of a semantic matching module in an MRC model; semantic matching preferred question types and passage types. Our work attempts to fill this gap and provide informative findings to the research community.

1 https://rajpurkar.github.io/SQuAD-explorer/
2 https://ai.google.com/research/NaturalQuestions/leaderboard
3. Comparison setups

We formulate a two-stage architecture to conduct the comparison and analysis. The architecture consists of two components: (1) a semantic matching module, which explicitly determines whether a piece of text is relevant to a question, and (2) a reading module, which reads a question and the semantic matching retrieved text, and generates candidate answers. We compare the effectiveness and efficiency from two aspects — different roles of semantic matching in MRC (i.e., reading-then-matching and matching-then-reading) in Section 3.1 and the different levels of semantic matching granularity (i.e., paragraph-level, sentence-level, and span-level matching) in Section 3.2. The overall comparison framework for semantic matching can be viewed as Fig. 2. Finally, we also introduce the methodology of the semantic matching and reading modules in Section 3.3.

3.1. Different roles of semantic matching

Different positions of a semantic matching module denote different roles in an MRC model. As shown in Fig. 3, in a matching-then-reading framework, the semantic matching module aims at selecting context for the following reading module. In a reading-then-matching framework, the semantic matching module aims at evaluating the candidate answers given by the reading module. We evaluate these two frameworks in the later experiments:

- **Matching-then-reading**: Semantic matching is used to filter out noisy context in this framework. Each input text is measured by a semantic matching model in terms of its relevance to the question. Irrelevant text is filtered out if its score is smaller than a threshold $t_{mr}$. The remaining pieces of text are combined and fed into a reading module, generating the final answer by the reading module.

- **Reading-then-matching**: Semantic matching is used to rank candidate answers, according to their matching scores to the question. The original text is fed into a reading module which generates the top-$n$ candidate answer spans. Then, for each span, the semantic matching module evaluates the relevance score of the context to the question. The final answer is given by the most relevant candidate answer.

3.2. Different levels of matching granularity

The different levels of granularity in context, such as paragraph, sentence, and span have different features in capturing the meaning of context (Mulkar-Mehta, Hobbs, & Hovy, 2011; Yin & Schütze, 2015). We explore the effect of semantic matching at different levels of granularity. More specifically, the input text evaluated by a semantic matching module falls into three categories:

- **Paragraph-level**: Input text is split into paragraphs. Then, the semantic matching module is trained to evaluate the relevance between a paragraph and a question, which will be leveraged to filter out noisy paragraphs in the matching-then-reading architecture or re-ranking candidate answers in the reading-then-matching architecture. Several existing works (Lee, Yun, Kim, Ko, & Kang, 2018; Wang, Yu, Guo, et al., 2018) show paragraph-level semantic matching is beneficial to MRC.
A piece of context for answering a question.

A special token, defined by a PLM.

An input question.

A special token, defined by a PLM.

A piece of context for answering a question.

Transformer hidden states.

Trainable parameters (bias).

Trainable parameters.

A Sigmoid function.

The number of training examples.

The probability of the $i$th training example.

The ground-truth labels of the $i$th token being at the start and the end positions, respectively.

The predicted probabilities of the $i$th token being at the start and the end positions, respectively.

The context for answering a question.

Transformer hidden states.

Input text is split into sentences. Then, the semantic matching module is trained to evaluate the relevance between the sentences and the question. Sentence-level learning can also be used for filtering (matching-then-reading) and re-ranking (reading-then-matching) paradigms. Several works (Choi, et al., 2017; Min et al., 2018; Nie, Chen, & Bansal, 2019) demonstrate the effectiveness of this approach.

This is usually used in the reading-then-matching architecture, where candidate answer spans generated by a reading module are evaluated in terms of their relevance to the question. Some recent works (Hu, Peng, et al., 2019; Kratzwald et al., 2019) illustrate that span-level answer re-ranking substantially improves MRC.

### 3.3. A comprehensive list of various setups

We use PLM as an encoder for both semantic matching and reading modules. To conduct a comparison of seven different settings of employing semantic matching to MRC, (1) M-RPara denotes the model follows a matching-then-reading framework and a matching on paragraph-level paradigm; (2) M-RSent denotes the model follows a matching-then-reading framework and a matching on sentence-level paradigm; (3) M-R(Para&Sent) denotes the model follows a matching-then-reading framework and a matching on sentence-level and paragraph-level paradigm; (4) R-MPara denotes the model follows a reading-then-matching framework and a matching on paragraph-level paradigm; (5) R-MSent denotes the model follows a reading-then-matching framework and a matching on sentence-level paradigm; (6) R-M(Para&Sent) denotes the model follows a reading-then-matching framework and a matching on sentence-level and paragraph-level paradigm; (7) R-MSpan denotes the model follows a reading-then-matching framework and a matching on span-level paradigm. These models are summarized in Table 2. We detail the algorithms of the semantic matching module and the reading module in the following subsections. The used symbols and their descriptions are summarized in Table 3.

#### 3.3.1. Semantic matching module

For a piece of context, i.e., a paragraph, a sentence, or a span, whether it matches to the question or not is defined as a binary classification problem in the semantic matching module. First, we generate its question-aware representation. The concatenated question and context with special tokens, e.g., “a special token [CLS], the question $Q$, a special token [SEP], and the context $t$” is fed into a PLM (RoBERTa-base in our implementation). Following Devlin et al. (2019), the representation of the [CLS] token is used as the aggregated sequence representation:

$$h_i = \text{Transformer}([CLS]; Q; [SEP]; t; [SEP])_{\text{cls}}, \tag{1}$$
where $\text{Transformer}$ denotes the PLM and $\mathbf{h} \in \mathbb{R}^d$ ($d$ denotes the dimension) is the question-aware representation of the context. Then, $\mathbf{h}$ is fed into a fully connected layer for classification:

$$p = \sigma(\mathbf{W}_c \cdot \mathbf{h} + b_c),$$

(2)

where $\sigma(\cdot)$ denotes the Sigmoid function, the scalar $p$ is the probability that the context is related to answer the question, and $\mathbf{W}_c$ and the bias $b_c$ are trainable parameters which are updated by minimizing the binary cross-entropy loss:

$$L_e = -\frac{1}{N} \sum_{i=1}^{N} y_i \log(p_i) + (1 - y_i)\log(1 - p_i),$$

(3)

where $N$ denotes the number of training examples, $p_i$ is the probability of $i$th example, and $y_i$ denotes the ground-truth label of $i$th example which is set to 1 when the context contains the answer to the question.

In the inference process, all pieces of context are re-ranked according to their probabilities (defined in Eq. (2)). The most relevant pieces of context are selected to answer the question using a threshold. We denote the context which contains the correct answer as golden context. The threshold is set according to the recall of golden context in the development dataset.

3.3.2. Reading module

The implementation of the reading module is based on PLMs. First, the question and the input context are concatenated in line with the method that was mentioned in the semantic matching module (see Section 3.3.1). Similarly, we extract the representations of all tokens from a PLM (e.g., BERT-base or RoBERTa-large):

$$\mathbf{H} = \text{Transformer}([\text{CLS}]; Q; [\text{SEP}]; T; [\text{SEP}]),$$

(4)

where $Q$ is the question, $T$ is the context for answering the question, $\mathbf{H} \in \mathbb{R}^{l \times d}$ is the representation matrix of all tokens, where $l$ is the size of tokens in the input sequence and $d$ is the dimension.

The reading model is trained to predict the start and end positions of the answer span. The model predicts the probability of each input token in terms of its likelihood of being the start token of the answer span:

$$\mathbf{p}^{(st)} = \text{softmax}(\mathbf{W}^{(st)} \cdot \mathbf{H} + b^{(st)}),$$

(5)

where $\mathbf{W}^{(st)}$ and the bias $b^{(st)}$ are trainable parameters, softmax is the Softmax function, and $\mathbf{p}^{(st)} \in \mathbb{R}^l$ is the distribution of probabilities of answer start position, where $l$ is the length of the input sequence. Likewise, the model also predicts the probability of each input token in terms of being the end token of the answer span:

$$\mathbf{p}^{(ed)} = \text{softmax}(\mathbf{W}^{(ed)} \cdot \mathbf{H} + b^{(ed)}),$$

(6)

where $\mathbf{W}^{(ed)}$ and the bias $b^{(ed)}$ are trainable parameters, and $\mathbf{p}^{(ed)} \in \mathbb{R}^l$ is the distribution of the probabilities of answer end position.

In the training stage, the training objective is to minimize the following cross-entropy loss:

$$L_s = -\frac{1}{N} \sum_{i=1}^{N} \sum_{j=1}^{l} y_j^{(st)} \log p_j^{(st)} + y_j^{(ed)} \log p_j^{(ed)},$$

(7)

where $N$ is the size of the training examples, $l$ is the length of the input sequence, $y_j^{(st)}$ and $y_j^{(ed)}$ are the ground truth of whether $i$th token is the start and end positions of the answer span, and $p_j^{(st)}$ and $p_j^{(ed)}$ are the probabilities that the $i$th token of being the start and end positions computed by Eqs. (5) and (6), respectively.

In the inference stage, all candidate answer spans are ranked by summing their probabilities of start and end positions:

$$\text{score}(A[m,n]) = p_m^{(st)} + p_n^{(ed)},$$

(8)

where $A[m,n]$ denotes a candidate answer span, $m$ and $n$ are the start and end positions, $p_m^{(st)}$ is the probability of the $m$th token being the start position, and $p_n^{(ed)}$ is the probability of the $n$th token being the end position. The candidate answer span with highest score is used to answer the question.

4. Experiments

4.1. Datasets

The experiments are conducted on four types of well-established datasets. The statistics of the used datasets are in Table 4. Details of the used datasets are as follows:

- **Open-SQuAD** is an open-domain dataset. The input context for each question is multiple documents from the entire Wikipedia domain (Chen et al., 2017). In our experiments, the context for answering each question is generated by retrieving the top-200 paragraphs according to TF–IDF scores. Two evaluation metrics are used: exact string match (EM) and F1 score, which measures the weighted average of precision and recall at the token level.
Table 4
The characterization of different MRC datasets. #Train and #Test denotes the number of questions in the training and testing datasets. #Paragraph, #Sentence and #Word denote the average number of paragraphs, sentences, and words for answering a question, respectively. Type denotes the types of context for answering the questions.

<table>
<thead>
<tr>
<th>Datasets</th>
<th>Type</th>
<th>#Train</th>
<th>#Test</th>
<th>#Paragraph</th>
<th>#Sentence</th>
<th>#Word</th>
</tr>
</thead>
<tbody>
<tr>
<td>Open-SQuAD</td>
<td>Open-domain Corpus</td>
<td>87.6k</td>
<td>10.6k</td>
<td>200</td>
<td>943.8</td>
<td>26.6k</td>
</tr>
<tr>
<td>NaturalQ</td>
<td>Long document</td>
<td>307.4k</td>
<td>7.8k</td>
<td>38.4</td>
<td>154.2</td>
<td>6.9k</td>
</tr>
<tr>
<td>SQuAD-2.0</td>
<td>Paragraph</td>
<td>130.3k</td>
<td>11.9k</td>
<td>1</td>
<td>4.4</td>
<td>127.9</td>
</tr>
<tr>
<td>AddOneSent</td>
<td>Adversarial paragraph</td>
<td>130.3k</td>
<td>1.8k</td>
<td>1</td>
<td>4.6</td>
<td>131.8</td>
</tr>
<tr>
<td>AddSent</td>
<td>Adversarial paragraph</td>
<td>130.3k</td>
<td>3.6k</td>
<td>1</td>
<td>4.9</td>
<td>134.8</td>
</tr>
</tbody>
</table>

Table 5
Performance of the reading module on different datasets. We report Long-F1 for Natural Questions and F1 for other datasets. The best performance is marked in bold and the worst performance is underlined.

<table>
<thead>
<tr>
<th>PLM</th>
<th>Open-SQuAD</th>
<th>Natural Questions</th>
<th>SQuAD-2.0</th>
<th>AddOneSent</th>
<th>AddSent</th>
</tr>
</thead>
<tbody>
<tr>
<td>BERT-base</td>
<td>37.13</td>
<td>59.48</td>
<td>76.58</td>
<td>78.71</td>
<td>72.72</td>
</tr>
<tr>
<td>BERT-large</td>
<td>39.27</td>
<td>65.59</td>
<td>81.80</td>
<td>84.86</td>
<td>79.64</td>
</tr>
<tr>
<td>RoBERTa-base</td>
<td>40.71</td>
<td>60.55</td>
<td>82.91</td>
<td>78.42</td>
<td>70.49</td>
</tr>
<tr>
<td>RoBERTa-large</td>
<td>42.74</td>
<td>67.36</td>
<td>88.56</td>
<td>87.35</td>
<td>83.66</td>
</tr>
</tbody>
</table>

- **Natural Questions** is a large-scale dataset where questions are collected from the Google search engine (Kwiatkowski et al., 2019). Each example comprises a real user question, a corresponding Wikipedia document, and two types of annotated answers: a long answer which is a passage (or a table, a list) containing all the information required to infer the answer; a short answer which is one or more entities which briefly answers the question. We use its official metrics for evaluation, i.e., Long-F1, Long-P, Long-R denote the F1, precision, and recall for long answers, and Short-F1, Short-P, Short-R denote the F1, precision, and recall for short answers.

- **SQuAD-2.0** is a popular MRC dataset where each example comprises with a question and a short paragraph (Rajpurkar et al., 2018). It contains 100,000 answerable questions from the work of Rajpurkar et al. (2016) and over 50,000 human crafted unanswerable questions. The evaluated metrics are EM and F1 on both the answerable subset and unanswerable subset.

- **Adversarial-SQuAD** is a challenging MRC dataset, where the evidence text for each question is a paragraph with auto-generated adversarial distracting sentences (Jia & Liang, 2017). Adversarial-SQuAD consists of two versions. **AddSent** adds grammatical sentences that look similar to the question, while the added sentences do not actually contradict the correct answers; **AddOneSent** adds a random human-approved sentence to each paragraph. The evaluated metrics are also EM and F1.

4.2. Implementation

We detail the implementation of the semantic matching module and reading module. Both modules are based on PLMs, and the two most commonly used PLMs are employed in this work, i.e., BERT (Devlin et al., 2019) and RoBERTa (Liu et al., 2019).

4.2.1. Semantic matching module

To generate the training data of the semantic matching module: for a weakly supervised dataset, i.e., Open-SQuAD, where only the answer span is provided to each question, all paragraphs and sentences containing the answer are labeled as positive, and the remaining ones are labeled as negative; for other strongly supervised datasets whose correct answers are labeled, we assign the paragraphs and sentences where contain the correct answers as positive, and the remaining as negative. We down sample the negative examples to keep the positive–negative ratio within 1:10.

The principle for setting parameters in the semantic matching module is to achieve high recall of golden paragraphs/sentences. The semantic matching module is built on RoBERTa-base because it achieves a balance between efficiency and effectiveness. The experiments show that stronger PLMs, such as BERT-large and RoBERTa-large, are able to achieve a better performance in semantic matching (i.e., 1.43% and 1.97% recall improvements on average), however they take much longer time in the training and inference stages. RoBERTa-base is a practical choice for semantic matching.

The semantic matching model is fine-tuned with two epochs with the following parameters: learning rate is 3e-5; batch size is 64; the maximum of sequence length is 384 and 128 for paragraph-level and sentence-level, respectively. The average length of paragraphs and sentences is 133 and 27, respectively, thus we set the sequence length as 384 and 128 to ensure most paragraphs/sentences are tokenized without truncations while keeping the model efficient. The matching module in the M-R framework selects a set of paragraphs or sentences by thresholding the scores, where the threshold $t_m$ is a hyperparameter. Similar to Min et al. (2018), we adjust $t_m$ according to the recall of golden paragraphs and sentences in the development sets, which is 65% recall for Open-SQuAD, 85% recall for Natural Questions, and 95% recall for other datasets.
module helps to remove most of the irrelevant paragraphs and makes it easy for the reading module to extract correct answers. The improvements on Open-SQuAD and Natural Questions are much larger than the improvements on SQuAD-2.0. A semantic matching substantially improves the MRC task over most of the datasets. More specifically,

5.1. RQ1: Does semantic matching improve MRC performance in all cases?

We compute the average performance of all settings (as detailed in Table 2) with semantic matching, and compute the improvements over their corresponding baseline methods without semantic matching. As shown in Fig. 4, it is observed that semantic matching substantially improves the MRC task over most of the datasets. More specifically,

(1) The contribution of semantic matching is particularly noticeable when there are many noisy paragraphs. For example, the improvements on Open-SQuAD and Natural Questions are much larger than the improvements on SQuAD-2.0. A semantic matching module helps to remove most of the irrelevant paragraphs and makes it easy for the reading module to extract correct answers. The more noisy paragraphs there are in an original dataset, the larger the improvement the MRC model achieves.
Table 7
Overall performance of different settings on Open-SQuAD and Natural Questions. We also compared them with published Rank 1-3 methods. For Open-SQuAD, they are Fusion-in-Decoder (Izacard & Grave, 2021), Path Retriever (Asai, Hashimoto, Hajishirzi, Socher, & Xiong, 2020), and Multi-Passage BERT (Wang, Ng, Ma, Nallapati, & Xiang, 2019). For Natural Questions, they are PoolingFormer (Zhang, Gong, et al., 2021), Cluster-Former (Wang, Zhou, et al., 2021), and RikiNet (Liu, Gong, et al., 2020).

<table>
<thead>
<tr>
<th>Datasets</th>
<th>Open-SQuAD</th>
<th>Natural Questions</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>EM</td>
<td>F1</td>
</tr>
<tr>
<td>RoBERTa-large Reader</td>
<td>Baseline</td>
<td>36.12 42.74</td>
</tr>
<tr>
<td>M-RPara</td>
<td>50.94 58.52</td>
<td>71.99 70.98</td>
</tr>
<tr>
<td>M-RSent</td>
<td>50.57 58.27</td>
<td>70.94 71.35</td>
</tr>
<tr>
<td>M-R(Para&amp;Sent)</td>
<td>38.87 45.87</td>
<td>70.23 71.13</td>
</tr>
<tr>
<td>R-MPara</td>
<td>40.17 46.86</td>
<td>71.00 71.80</td>
</tr>
<tr>
<td>R-MSent</td>
<td>39.79 46.51</td>
<td>70.77 72.07</td>
</tr>
<tr>
<td>R-M(Para&amp;Sent)</td>
<td>39.46 46.33</td>
<td>67.85 69.53</td>
</tr>
<tr>
<td>R-MSpan</td>
<td>37.74 44.59</td>
<td>67.50 66.87</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Datasets</th>
<th>SQuAD-2.0</th>
<th>AddOneSent</th>
<th>AddSent</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>EM</td>
<td>F1</td>
<td>HasAns</td>
</tr>
<tr>
<td>RoBERTa-large Reader</td>
<td>Baseline</td>
<td>85.29 88.56 82.49</td>
<td>89.04 88.07</td>
</tr>
<tr>
<td>M-RSent</td>
<td>84.84 88.14 80.63</td>
<td>87.24 89.03</td>
<td>83.10</td>
</tr>
<tr>
<td>R-MSent</td>
<td>84.36 88.01 80.99</td>
<td>88.30 87.72</td>
<td>81.09</td>
</tr>
<tr>
<td>R-MSpan</td>
<td>85.34 88.51 82.42</td>
<td>88.78 88.24</td>
<td>81.53</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Datasets</th>
<th>State-of-the-art</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Rank 1</td>
</tr>
<tr>
<td></td>
<td>56.7</td>
</tr>
<tr>
<td></td>
<td>63.2</td>
</tr>
<tr>
<td></td>
<td>79.82</td>
</tr>
<tr>
<td></td>
<td>78.47</td>
</tr>
<tr>
<td></td>
<td>82.22</td>
</tr>
<tr>
<td></td>
<td>61.63</td>
</tr>
<tr>
<td></td>
<td>70.37</td>
</tr>
<tr>
<td></td>
<td>58.42</td>
</tr>
</tbody>
</table>

(2) Semantic matching increases the robustness of the reading model against adversarial attack. For example, for AddOneSent and AddSent, using a semantic matching module improves RoBERTa-large and BERT-base baselines by the average F1 scores of 1.7 and 3.3, respectively. The improvements are mainly due to the removal of adversarial sentences by the semantic matching module, i.e., 74.3% and 74.9% of adversarial sentences are filtered out from AddOneSent and AddSent, respectively. We labeled the adversarial sentences added by humans by comparing them with the original evidence text in SQuAD-1.1 (Rajpurkar et al., 2016) and evaluated them using the sentence-level matching module. We randomly selected error cases of the baseline method (i.e., RoBERTa-large) which are predicted with wrong answers. As shown in Table 9, the predicted answers are located in adversarial sentences, showing that the baseline method is distracted by attacking information. We also report the answers predicted by our method (M-RSent). It is observed that it is easy to identify adversarial sentences using sentence-level semantic matching, hence the reader correctly answers the questions.
Fig. 4. The average F1 improvements of different settings over baseline methods (i.e., RoBERTa-large and BERT-base) on different datasets.

Table 9
Example questions from AddOneSent and AddSent. Human-crafted sentences are marked with ✓ in the Adversarial column. Sentences which are detected as being semantically matched to the question are marked with ✓ in the Match column, otherwise they are marked with ×. The ground truth span is underlined. The prediction from our method (M-RSent) is marked in bold in blue. The prediction from the RoBERTa-large baseline is marked in italics in red.

<table>
<thead>
<tr>
<th>Adversarial</th>
<th>Match</th>
<th>Question</th>
<th>Prediction</th>
</tr>
</thead>
<tbody>
<tr>
<td>×</td>
<td>✓</td>
<td>On what date was Super Bowl 50 given to Levi’s Stadium?</td>
<td>On May 21, 2013, NFL owners at their spring meetings in Boston voted and awarded the game to Levi’s Stadium.</td>
</tr>
<tr>
<td>✓</td>
<td>×</td>
<td>On the date of November 10, 1988 was Champ Bowl 40 given to Klein’s Stadium.</td>
<td>On the date of November 10, 1988 was Champ Bowl 40 given to Klein’s Stadium.</td>
</tr>
<tr>
<td>×</td>
<td>✓</td>
<td>Who filed an objection to the BBC using the blue police box in Doctor Who merchandise?</td>
<td>In 1998, the Metropolitan Police Authority filed an objection to the trade mark claim; but in 2002, the Patent Office ruled in favor of the BBC.</td>
</tr>
<tr>
<td>✓</td>
<td>×</td>
<td>What is the unproven assumption generally ascribed to the value of complexity classes?</td>
<td>Jeff Dean filed an objection to the ITV for using its merchandise, the blue police box in Doctor Who.</td>
</tr>
</tbody>
</table>

Fig. 5. A comparison of the effectiveness of different matching granularity. BL is the baseline model, and P&S denotes Para&Sent. The average F1 scores are reported.

(3) We observe that semantic matching only yields a moderate performance on SQuAD-2.0 where evidence text is a concise paragraph. The reason is two-fold. First, the large-scale PLM is a powerful classifier, outperforming human performance on this dataset (as shown in the leaderboard⁴). Second, the dataset was constructed by proposing a question based on a given paragraph. The questions and evidence text had been strongly matched. Hence, there is little room for improvement using a semantic matching module to select evidence text.

(4) Semantic matching achieves greater gains on a weak PLM than a strong one. For example, it improves the vanilla RoBERTa-large baseline by 2.4% and the vanilla BERT-base baseline by 4.06% on average over the four datasets. Moreover, as shown in Table 8, the M-RPara model with BERT-base reader achieves 53.61 F1 score and surpasses the RoBERTa-large baseline (42.74 F1 score) by a large margin on Open-SQuAD.

5.2. RQ2: Which granularity of semantic matching is more effective?

Fig. 5 summarizes the average F1 performance of different matching granularity across the four datasets. We make the following observations. First, paragraph-level matching is slightly better than sentence-level matching, however combining them does not result in additional improvements. There are two possible reasons for the superiority of paragraph-level matching. On the one hand, it is easy for sentence-level semantic matching to filter out visibly noisy sentences, but difficult for it to distinguish plausible sentences, whereas the paragraph-level matching provides more accurate and differentiated information to avoid wrong predictions. On the other hand, sentence-level matching selects sentences from different paragraphs.

---

*https://rajpurkar.github.io/SQuAD-explorer/*
Table 10

Example from the Natural Questions dataset. We list three paragraphs. The first paragraph (P1) is detected as semantically matched to the question, while the other two paragraphs are not. Sentences in bold are detected as semantically matched to the question. The ground truth span is underlined. The prediction from our method is marked in blue. The prediction from baseline method (i.e., RoBERTa-large) is marked in red.

**Question:** When did Hollywood become the centre of the film industry?

<table>
<thead>
<tr>
<th>P1 match</th>
<th>By 1912, major motion - picture companies had set up production near or in Los Angeles. In the early 1900s, most motion picture patents were held by Thomas Edison’s Motion Picture Patents Company in New Jersey. To escape this, filmmakers began moving out west, where Edison’s patents could not be enforced. Also, the weather was ideal and there was quick access to various settings. Los Angeles became the capital of the film industry.</th>
</tr>
</thead>
</table>
| P2 not match | Director D.W. Griffith was the first to make a motion picture in Hollywood. His 17-minute short film In Old California (1910) was filmed for the Biograph Company. Although Hollywood banned movie theaters ... before annexation that year, Los Angeles had no such restriction. The first film by a Hollywood studio, Nestor Motion Picture Company, was shot on Oct 26, 1911. The H. J. Whitley home was used as its set, and the unnamed movie was filmed in the middle ...
| P3 not match | The first studio in Hollywood, the Nestor Company, was established by the New Jersey – based Centaur Company in a roadhouse at 6121 Sunset Boulevard (the corner of Gower), in Oct 1911. Four major film companies...had studios in Hollywood, as did several minor companies ... In the 1920s, Hollywood was the fifth-largest industry in the nation. |

As a result, the concatenated context that is fed into a reading model will become incoherent. As shown in Table 10, we randomly selected a question from the Natural Questions dataset which was answered correctly by M-RPara, but was answered incorrectly by M-RSent. Taking this question as an example, only the bold sentences are selected by a sentence-level semantic matching module. They are then concatenated and fed into the reading model. However, the bold sentences scatter in different paragraphs and talk about different topics. Thus, it is difficult to generate a correct answer from them.

Second, both paragraph-level and sentence-level matching methods perform better than span-level matching. This is mainly because the fine-grained semantic matching text does not contain sufficient contextual information that can answer the question. For example, as shown in Table 10, for the question “When did Hollywood become the centre of the film industry?”, the matching sentence “Los Angeles became the capital of the film industry” is at the end of the paragraph, whereas the correct answer span is “1912” at the beginning of the paragraph. In the sentence-level matching setup, the sentence containing the answer span is filtered out by the semantic matching module, thus yielding an incorrect result.

Finally, the span-based methods yield weaker results than the sentence-based and paragraph-based methods in the reading-then-matching framework. In this framework, the span-based semantic matching module evaluates the relevance of the identified spans to the question. The weaker performance also shows that coarse-grained context, e.g., sentence-level and paragraph-level can help the semantic matching module better rank the retrieved candidate context in MRC.

5.3. RQ3: Matching first or reading first?

We compare both the effectiveness and efficiency of the different roles of semantic matching in MRC, i.e., matching-then-reading and reading-then-matching.

(1) We compare the average F1 performance of the different frameworks in Fig. 6. The matching-then-reading framework achieves greater gains than the reading-then-matching over the baseline models across all datasets. A possible reason is that a reading model is prone to be distracted by irrelevant or adversarial context, e.g., paragraphs or sentences. Therefore, in the matching-then-reading framework, it is easier for a reading model to generate correct answers after irrelevant or adversarial context has been filtered out. In contrast, in the reading-then-matching framework, it is likely that top candidate answers generated by the reading model do not contain a correct answer. According to our experiments, over 25% examples do not contain correct answers in the top-20 candidate spans on the Natural Questions dataset. In these cases, the system will generate incorrect answers, no matter how well the matching model performs.
Table 11
The comparison of the efficiency of different frameworks on Open-SQuAD and Natural Questions datasets. avg.n refers to the average number (per example) of paragraphs or sentences fed into the matching module. total refers to the whole framework, where time refers to the time consumed, and speed is calculated as dividing the time cost of the vanilla RoBERTa-large baseline model by the time cost of our model.

<table>
<thead>
<tr>
<th>Matching granularity</th>
<th>Datasets</th>
<th>Matching-then-reading</th>
<th></th>
<th>Reading-then-matching</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Avg.n</td>
<td>Time</td>
<td>Avg.n</td>
<td>Time</td>
</tr>
<tr>
<td>Paragraph</td>
<td>Open-SQuAD</td>
<td>200</td>
<td>3.25 h</td>
<td>3</td>
<td>58 m</td>
</tr>
<tr>
<td></td>
<td>NaturalQ</td>
<td>38.4</td>
<td>3.08 h</td>
<td>1.7</td>
<td>28 m</td>
</tr>
<tr>
<td>Sentence</td>
<td>Open-SQuAD</td>
<td>943</td>
<td>4.75 h</td>
<td>20</td>
<td>41 m</td>
</tr>
<tr>
<td></td>
<td>NaturalQ</td>
<td>154.2</td>
<td>1.50 h</td>
<td>29.7</td>
<td>44 m</td>
</tr>
</tbody>
</table>

Table 12
Proportion of predicted answers which are located in the wrong sentences in SQuAD-2.0 and Open-SQuAD, and the adversarial sentences in AddOneSent and AddSent. +Matching denotes we add a semantic matching module, following an M-RSent paradigm. The lower the rates, the better.

<table>
<thead>
<tr>
<th>Methods</th>
<th>Vanilla-RoBERTa_{large}</th>
<th>+Matching</th>
<th>Δ</th>
</tr>
</thead>
<tbody>
<tr>
<td>SQuAD-2.0</td>
<td>2.21%</td>
<td>2.45%</td>
<td>+0.24%</td>
</tr>
<tr>
<td>Open-SQuAD</td>
<td>53.64%</td>
<td>48.49%</td>
<td>−5.15%</td>
</tr>
<tr>
<td>AddOneSent</td>
<td>10.18%</td>
<td>4.81%</td>
<td>−5.37%</td>
</tr>
<tr>
<td>AddSent</td>
<td>16.10%</td>
<td>6.99%</td>
<td>−9.10%</td>
</tr>
</tbody>
</table>

(2) We compare the time cost in the inference phase of these two frameworks in Table 11. As shown in the table, the matching-then-reading framework with paragraph-level matching is more efficient (3.1 times faster), compared with the vanilla RoBERTa-large baseline model on the Open-SQuAD dataset. This is mainly because the reading model only needs to read three paragraphs after the matching module. In contrast, the reading-then-matching framework takes more time than the baseline model. Similar patterns can also be observed in the sentence-level matching-based framework.

To sum up, the matching-then-reading framework outperforms the reading-then-matching framework in terms of both effectiveness and efficiency.

5.4. RQ4: What types of passages are more suitable for introducing a semantic matching module?

First, we classify the input text for question answering into three types: (i) noisy: multiple passages from open-domain text or a long Wikipedia document, i.e., Open-SQuAD, (ii) concise: a question-related passage, i.e., SQuAD-2.0, and (iii) adversary: a passage with adversarial sentences, i.e., AddOneSent and AddSent. We conduct experiments using the matching-then-reading framework, and a sentence-level semantic matching model is employed to filter out noisy sentences from the input text. We do not test the paragraph-level matching model because it does not support the experiments on SQuAD-2.0, AddOneSent and AddSent datasets (see Section 4.2). In Table 12, we compare the proportion of examples which extract answers from wrong sentences in SQuAD-2.0, AddOneSent and AddSent datasets. When simply using RoBERTa-large for the reading model, 16.1% of the examples in AddSent are distracted by the adversarial information. After using a semantic matching module, this proportion drops to 6.99%. This suggests that the semantic matching model can resist adversarial attacks. For the Open-SQuAD, the semantic matching module reduces distraction from wrong sentences by 5.15%, showing its effectiveness under noisy settings. Only 2.21% of examples in SQuAD-2.0 are distracted by the wrong sentences when the original text is fed into the reading module only. We observe that introducing a semantic matching model slightly increases the error rate by 0.24%, indicating that it is not suitable for the concise MRC task.

5.5. RQ5: What types of questions are more suitable for introducing a semantic matching module?

On the Adversarial-SQuAD dataset, we analyze which types of questions are more susceptible to adversarial information. We also compare a vanilla RoBERTa-large baseline model and a model with an additional M-RSent-based semantic matching module. The examined question types include what, where, when, how, why, who and which questions. For each type, the proportion of examples which generate answers from the adversarial sentences (incorrect) is shown in Fig. 7. We observe that the semantic matching module achieves significant improvements on what, where, when, how, who and which questions, however it underperforms on why questions. Next, to analyze these questions, we randomly select different types of questions from AddOneSent and AddSent datasets as shown in Table 13. We observe that the semantic matching module is very useful in relation to questions which can be answered using one sentence as evidence. For the degradation caused by using a semantic matching module, a potential reason is that multiple sentences are required to answer the question.

---

4 We compare the text in SQuAD-Adversarial with its original text in SQuAD-1.1 to identify the adversarial sentences.
Fig. 7. Percentage of the predicted answers which are located in the adversarial sentences in AddOneSent (left) and AddSent (right). Vanilla denotes the RoBERTa-large baseline model without a semantic matching module. +Matching denotes we add a semantic matching module, following an M-RSent paradigm. The lower the rate, the better.

Table 13

<table>
<thead>
<tr>
<th>Examples of different question types. The questions are in bold and the answers are underlined.</th>
</tr>
</thead>
</table>
| **Who** is the chair of the IPCC?  
TKorean economist Hoesung Lee is the chair of the IPCC since October 8, 2015. |
| **When** did operation anvil open?  
Operation Anvil opened on 24 April 1954, after weeks of planning by the army with the approval of the War Council. |
| **Where** was the famous artist Tamara de Lempicka born?  
Tamara de Lempicka was a famous artist born in Warsaw. |
| **What causes** rock extension?  
Extension causes the rock units as a whole to become longer and thinner. This is primarily accomplished through normal faulting and through ductile stretching and thinning. |
| **Why** was the rhine measurement changed?  
Until 1932 the generally accepted length of the Rhine was 1,230 kilometers (764 miles). In 1932 the German encyclopedia Knaurs Lexikon stated the length as 1,320 kilometers (820 miles), presumably a typographical error. The error was discovered in 2010, and the Dutch Rijkswaterstaat confirms the length at 1,232 kilometers (766 miles). |

For example, for the question “what causes rock extension”, the referential phenomena (i.e., This is referred to rock extension) requires combining multiple sentences to answer the question. Another cause is that the question is difficult and complex, e.g., for the question “why was the rhine measurement changed”, the sentence-level semantic matching module is unable to process changed, which required an in-depth language understanding and reasoning ability. Notably, only 3.0%, 3.3%, and 3.9% of questions in Open-SQuAD, SQuAD-2.0, and Adversarial-SQuAD are why questions. Min et al. (2018) also found that 90% of questions in the SQuAD dataset can be answered by one evidence sentence. These can explain the performance improvements of introducing a semantic matching module in MRC, because semantic matching helps to locate the only evidence sentence, and why questions are infrequent in these benchmarking datasets.

6. Conclusion

This paper is an empirical study of semantic matching for the MRC task. We systematically compare and analyze the performance of introducing a semantic matching module from two aspects, the different roles of semantic matching and the different granularity of matching context. The experiment results show that semantic matching is especially beneficial when there is noisy or adversarial context, or the reading module is relatively weak, e.g., using BERT-base for the reading module. We also find that the reading-then-matching framework with paragraph-level matching achieves the largest improvements. These findings may provide guidance for industrial applications and hopefully shed light on potentially advanced MRC techniques.

Based on our review, the limitations of employing semantic matching in MRC are two-fold. First, PLM-based semantic matching modules cannot address complex questions because they lack the capacity for deep semantic understanding. Second, the concatenation of question-related texts is not conducive to answer the question because they are extracted from different documents or paragraphs. The concatenation might change the original semantics. In the future, we would like to design a semantic matching module with an in-depth language understanding ability to handle complex questions. Moreover, we would like to investigate how to implement the semantic matching module in the MRC task efficiently and effectively.

CRediT authorship contribution statement

Qian Liu: Conceptualization, Methodology, Software, Writing – original draft. Rui Mao: Data curation, Writing – review & editing, Investigation. Xiubo Geng: Conceptualization, Writing – review & editing, Resources. Erik Cambria: Writing – review & editing, Formal analysis, Supervision.
Data availability

Data will be made available on request.

Acknowledgment

This research is supported by the Agency for Science, Technology and Research (A*STAR), Singapore under its AME Programmatic Funding Scheme (Project #A18A2b0046).

References


Ji, Yujie, Chen, Liangyu, Dou, Chenxiaio, Ma, Baochang, & Li, Xiangang (2022). To answer or not to answer? Improving machine reading comprehension model with span-based contrastive learning. In Findings of NAACL (pp. 1292–1300).


Liu, Qian, Geng, Xiubo, Huang, Heyan, Qin, Tao, Lu, Jie, & Jiang, Daxin (2021). MGRC: An end-to-end multigranularity reading comprehension model for question answering. *IEEE Transactions on Neural Networks and Learning Systems*, 1–12.


Nie, Yixin, Wang, Songhe, & Bansal, Mohit (2019). Revealing the importance of semantic retrieval for machine reading at scale. (pp. 2553–2566).


