HintMiner: Automatic Question Hints Mining From Q&A Web Posts with Language Model via Self-Supervised Learning

Zhenyu Zhang
JD AI

JiuDong Yang
JD AI

Abstract

Users often need ask questions and seek answers online. The Question - Answering (QA) forums such as Stack Overflow cannot always respond to the questions timely and properly. In this paper, we propose HintMiner, a novel automatic question hints mining tool for users to help them find answers. HintMiner leverages the machine comprehension and sequence generation techniques to automatically generate hints for users' questions. It firstly retrieve many web Q&A posts and then extract some hints from the posts using MiningNet that is built via a language model. Using the huge amount of online Q&A posts, we design a self-supervised objective to train the MiningNet that is a neural encoder-decoder model based on the transformer and copying mechanisms. We have evaluated HintMiner on 60,000 Stack Overflow questions. The experiment results show that the proposed approach is effective. For example, HintMiner achieves an average BLEU score of 36.17% and an average ROUGE-2 score of 36.29%. Our tool and experimental data are publicly available. [1].

1 Introduction

It is a common practice to seek answers from online Question and Answering (Q&A) forums, such as Stack Overflow, Data Science, etc. [Wang et al., 2018a, Chen et al., 2018, Calefato et al., 2018]. These Q&A forums store abundant question related posts accumulated over years. However, as the posted questions in Q&A sites rely on community members' voluntary answers, there is no guarantee to obtain timely and satisfactory answers for everyone question. As a matter of fact, we have found that a large number of questions lack accepted answers in Stack Exchange. It also costs users lots of time to search from those webs, where the Q&A resource aggregating and reforming methods are quite a necessity.

In recent years, some methods have been proposed to help users with Q&A. Some retrieval based methods such as AnswerBot [Xu et al., 2017] or the official Stack-Overflow website specify the key points of answers from the retrieved relevant posts by selecting the most important paragraphs. Another kind effective Q&A method is machine reading comprehension (MRC), which aims to understand the semantics of question and then select a text span from a given passage [Rajpurkar et al., 2018, Wang et al., 2018b, Wang et al., 2017, Chen et al., 2017] as the answer to the question. However, the MRC cannot combine several spans to form a more rich and semantic-complete result. Enlightened by the Q&A systems based on retrieval and MRC such as DrQA[Chen et al., 2017], etc., we build a dedicated automatic question hints mining system to help users. We targeted at mining hints from Q&A forums while these methods do not utilize the specific Q&A web resources. And we also try to merge several selected spans to generate semantic rich and complete results while those previous works can only retrieve passages or select independent text spans.

In this paper, we aim to reuse the existing resources in online Q&A forums to generate useful hints to the user questions. To that end, we propose a question hints mining tool called HintMiner, which selects and merges several useful segments of texts that can provide some hints for the question. We formulate HintMiner as: **Find the most useful text spans from the relevant posts in Q&A forums and combine them to generate the hints for the question.** Based on the hints provided, it will be much easier for users to get the final answers or help users to clarify

[1] `https://github.com/zhangzhenyu13/HintMiner`

Proceedings of the 27th International Conference on Artificial Intelligence and Statistics (AISTATS) 2024, Valencia, Spain. PMLR: Volume 238.

and understand the questions. HintMiner first leverages Elastic Search (**ES**[2]) to find relevant information for the question. It then selects several text-spans that can provide some hints for a question to form the answer through machine reading comprehension [Chen et al., 2017]. Finally, HintMiner merge the text-spans to generate semantic rich and complete hints via sequence generation [Ranzato et al., 2015]. To achieve this, we designed a self-supervised learning(SSL) objective for MiningNet to capture the semantics of questions and the relevant posts and to generate suitable hints. We construct "question" + "relative posts" + "proper hints/answers" triplets from millions of online stackoverflow posts. Then we train the MiningNet to learn to generate such "hints/answers" with "questions" + "relative posts" as input. MiningNet leverages BERT [Devlin et al., 2018] to encode the question and its relevant posts so as to capture their deep semantics. The deep semantic representation is further fed to a transformer decoder [Vaswani et al., 2017] that can capture the importance of each input token through the attention mechanism. With the learned token importance, we build a CopyNet using the copy mechanism [Gu et al., 2016, Zhou et al., 2018] to select a set of relevant tokens from the input to generate hints.

We have conducted extensive experiments to evaluate HintMiner. The results show that HintMiner outperforms several information retrieval based methods. For example, HintMiner achieves an average of 36.17% BLEU score and 36.29% ROUGE-2 score. Furthermore, MiningNet outperforms several strong retrieval baselines and generation language model baselines.

Our contributions can be summarized as follows:

- We build an automatic question hints mining tool called HintMiner, which can help developers solve questions. We extracted paragraphs from online Q&A forums to build a useful posts dataset. We also make our code and data publicly available.
- We develop MiningNet, a novel self-supervised learning based model that can capture the semantics of a question and the relevant posts in Q&A forums, and can generate the semantic rich and complete hints for questions.
- We have performed extensive evaluation of the proposed approach. Our results show that HintMiner is effective and outperforms several strong baseline methods.

Our work is an important step towards intelligent hints mining for Q&A forums.

[2] `https://www.elastic.co/elasticsearch/`

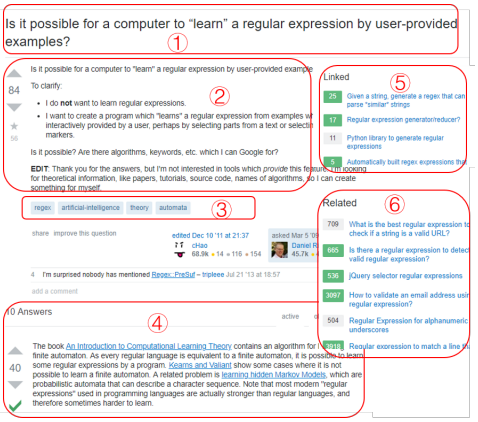

Figure 1: An example of the posts in Stack Overflow

2 The Q&A Forums and Dataset

2.1 Question Answering Web Resources

Web users would always ask questions or search relevant answers online. To solve their problems, all kinds of online users depend heavily on online Q&A sites. For example, Stack Overflow has become one of the most popular such Q&A sites for developers, and it has accumulated a large number (over 16 million) of Q&A posts. Figure 1 shows an example of the posts in Stack Overflow website. There are mainly six parts in a post: 1) the title of the question, showing the general concise description of the question, 2) the detailed description of the question, 3) the tags assigned by the user who posts the question, indicating the categories of the question, 4) the list of the answers to the question, including the accepted answer if available, 5) the linked posts that are marked by Stack Overflow community which are relevant to the current post, and 6) the related posts that are retrieved by the Stack Overflow system. It has been found that the responding time can be quite long and many questions may never be answered [Wang et al., 2018a]. It is desirable to improve question solution effectiveness by automating the question answering process. Therefore, it is quite necessity to find proper hints for users' questions.

2.2 The Construction of the Q&A Dataset

We collected about 17 million posts from four Stack Exchange websites via *archive.org*, including Stack Overflow[3], Artificial Intelligence[4], Data Science[5] and Cross Validated[6]. As illustrated in Figure 1, the linked posts are marked by the community and are useful

[3] https://stackoverflow.com/
[4] https://ai.stackexchange.com/
[5] https://datascience.stackexchange.com/
[6] https://stats.stackexchange.com/

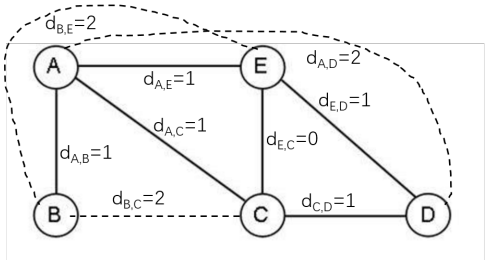

Figure 2: An Example of Post-Link Graph

to the question. There are about 19% of the posts connected with over $5M$ links. We build a Post-Link Graph where the nodes are posts and the edges are weighted links. There are two types of links marked by the community. We set the weights to 0 for links that mark duplicate posts and 1 for the others. Then we apply Dijkstra algorithm to compute the shortest link distance between each pair of nodes. Finally, we obtain 4 link distances ("0", "1", "2" and "≥3") because previous researches [Xu et al., 2016, Ye et al., 2017] show that two posts with a link distance $d \geq 3$ is not relevant to each other. For example, in Figure 2, there are 6 marked links between A, B, C, D, E and we complete the rest links (dashed lines) except for B, D as $d_{B,D} \geq 3$. The link distance indicates how useful the post content is to the question of the other post. The shorter the link distance is, the more useful the post is to the question.

2.2.1 Selecting Relevant Posts For Training

In order to train the MiningNet (Section 3.2), we build a "question-passage-hints" triplets dataset. For a question of the node (i.e. Post) in the post-link graph, we select top 2, 1, 1, 1 paragraphs for posts with distance as 0,1,2,and $\geq$ 3 respectively to construct the relevant passage of current question. The paragraphs of the passage are randomly shuffled so that the model cannot simply remember order of sentences. We select the first passage of accepted answer in the post with more than 10 words as the **gold hints** for the question, which is considered to be meaningful. We removed those questions without neighbors whose distance is 1. Finally we constructed about 3.6 million "question-passage-hints" triplets. Note that we add some less relevant paragraphs whose distance is larger than 1 so that noise and negative content are added to improve the robustness and difficulty of the dataset.

2.2.2 Selecting Relevant Posts For Inference

We first dumped the posts into the Elastic Search Engine (**ES**), and then retrieve relevant posts from a variety of Q&A forums. We then perform pre-processing of the selected posts. In this work, we aim at generating hints rather than generating code or numerical expressions which usually exists in those scientific forums. Therefore, we replace a code snippet with **[CODE]**, and a mathematical expression with **[NUM]**. We do not consider hyperlinks either. To reduce the vocabulary size, we use the BPE algorithm [Wu et al., 2016] to perform tokenization, which can transform a compound word into a few tokens. For each question, we retrieve 5 posts in total.

The retrieved posts often contain many non-essential sentences that are useless and can make it difficult for a deep neural network to handle extremely long input [Koehn and Knowles, 2017]. Examples of such sentences are "Maybe my answer can help you", "Thank you for your suggestion", etc. Therefore, we leverage an ensemble method to filter those sentence, which combines the results of three base algorithms that can identify the important sentences. 1) *Lexrank* [Erkan and Radev, 2004], a graph based method inspired by Pagerank algorithm[Wills, 2006], which uses the eigenvector centrality of sentences to select the important sentences. 2) *KL greedy search* [Haghighi and Vanderwende, 2009], an information entropy maximization based method, which uses the KL divergence to compute the relative information gain to greedily select sentences so as to maximize the information entropy of selected sentences. 3) *Latent Semantic Analysis (LSA)* [Steinberger and Jezek, 2004], which decomposes the sentence-term matrix using SVD and selects the sentences with the most significant topics via the right singular vectors. The three base algorithms focus on different aspects of sentence importance. Therefore, we merge their results and eliminate the sentence repetition. The resulting set of sentences forms the context passage for MiningNet.

3 HintMiner: Generating Hints to User's Questions

3.1 System Overview

In our work, we formulate the problem as follows: given a question and a set of relevant posts, the core problem is to select a set of useful text spans from existing posts and generate the hints to the question. We process the posts to form the context passage for training (Section 2.2.1) and inference (Section 2.2.2). The hints are then generated by the MiningNet.

For that purpose, we build HintMiner, which utilizes the techniques of machine reading comprehension [Chen et al., 2017] and sequence generation [Ranzato et al., 2015]. Figure 3 shows the overview

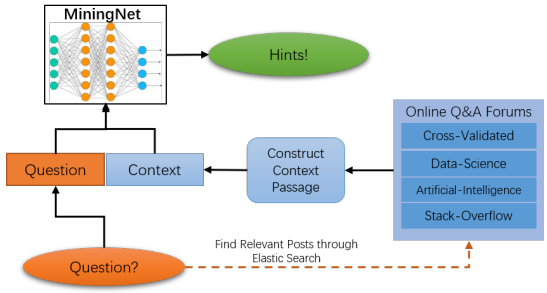

Figure 3: An Overview of HintMiner

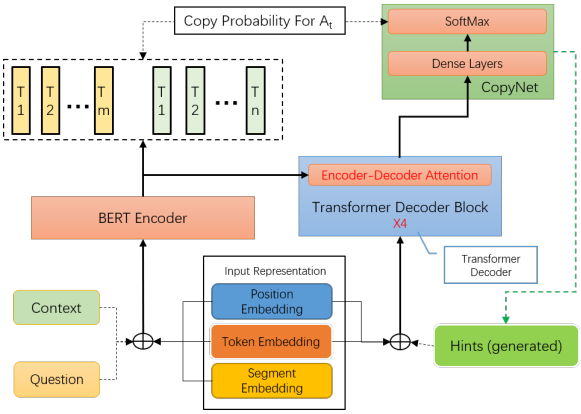

Figure 4: An Overview of MiningNet

of HintMiner. Given a question, we first select the relevant posts from the Q&A forums to form the context passage (Section 2.2.2). Then we feed the context and question to MiningNet, which is an effective deep neural network that can generate the hints to the question by copying and generating tokens from the context.

3.2 The MiningNet Model

3.2.1 The Structure of the Model

Figure 4 shows the structure of MiningNet, which consists of four parts: a BERT Encoder, a transformer decoder, and a CopyNet. There are three embeddings in the Input Representation that takes Q&A data as input and outputs the embeddings of input. These embeddings are extracted from BERT [Devlin et al., 2018]. It is worth mentioning that the segment id for tokens in questions is 0 and for tokens in context is 1. When generating the t^{th} answer token (A_t), the tokens of the generated answer before step t ($A_1, ..., A_{t-1}$) are embedded using position embedding and token embedding only, and are then fed to the transformer decoder. The $\oplus$ in Figure 4 refers to the use of BERT embedding layer to embed the tokens in text. Equation 1 presents how BERT is used to encode the sequence in our model. Each token in q (a question) and c (the context) is encoded as a dim dimension dense vector T_i^q/T_j^c, and the T_{cls} represents the pooling vector.

$$T = [T_{CLS}, T_1^q, , ..., T_m^q, T_1^c, ..., T_n^c] = BERT([q, c]]),$$
$$where\, T_i^q, T_j^c \in R^{dim},\ 1 \le i \le m,\ 1 \le j \le n \quad (1)$$

The output of the BERT encoder is a sequence of vectors representing the semantics of the question and context passage. The transformer decoder [Vaswani et al., 2017] reads the output of the BERT encoder and then computes the output hidden state of target (i.e. generated answer) vectors. The transformer decoder also computes the encoder-decoder attention score vectors, which uses the multi-head attention mechanism to pay attention to the "question+context passage". We also build a CopyNet [Gu et al., 2016, Zhou et al., 2018], which takes the encoder-decoder attention vectors as input and outputs an answer text. Using CopyNet, certain text spans in the input sequence are selected to be present in the output sequence with the Copy Probability [Gu et al., 2016]. Thus, MiningNet can generate the hints to a question through the copying mechanism by selectively replicating the input text spans. In this way, we transform the hints generation problem to a MRC problem, where the hints is composed of several selected text spans and combined via generation. The hints tokens A_1, A_2, ... A_n are generated one by one through the copying mechanism iteratively until A_n is the end token or n exceeds the hints length limit. (The generation length (n) is usually set to a fixed length for satisfactory model performance [Koehn and Knowles, 2017].)

3.2.2 Transformer Decoder and CopyNet

In this subsection, we describe the Transformer Decoder and CopyNet models in detail and show how to adapt them to MiningNet. The encoder vectors (T) are the semantic representation of "question+context". We apply the transformer decoder to them and compute the encoder and decoder attention via Equation 2, which defines the compatibility function of the query with the corresponding key in the multi-head attention. Q denotes the decoder hidden vectors at time step t, which is given as shifted masked $t-1$ true answer encoding vectors during training and $t-1$ predicted answer encoding vectors during testing.

To select text spans from the context, we apply the copying mechanism ($Copy$) on the encoder-decoder attention vectors, which can select tokens from input directly. According to the copy mechanism and the attention in Equation 2, the output of CopyNet (a three layer MLP with same model dimension and activation function used in BERT) is

denoted as $p(A_t|A_1, ..., A_{t-1}, T) = Copy(attn^t) = softmax(attn^t)$, where $attn^t_j$ is the attention value at step t for j^{th} context vector. Therefore, the probability distribution for generated answer sequence $p(A_1), p(A_2), ..., p(A_n)$ can be denoted as Equation 3.

$$Attention(Q, K, V) = softmax(\frac{Q * K^T}{\sqrt{dim}}) * V, \quad where\, K = V = BERT([q, c]) \tag{2}$$

$$p(A|T) = \prod_t p(A_t|A_1, ..., A_{t-1}, T) \tag{3}$$

It is worth mentioning that the hints generation process is based on captured semantics of question and context (passage formed from relevant posts). Essentially, MiningNet simulates a function that maps the semantic representation of the **context to hints** text spans according to the **requirement of the question** via the attention mechanism. The context contains the knowledge that can provide some hints for the question, which is represented as encoded vectors. The BERT encoder leverages its well designed and pre-trained network to represent the semantics of the question and context, then the decoder computes an attention score for each token of hints that is to be copied from the context based on the semantics. In the end, the selected text-spans could represent the most proper hints that can help to clarify and understand the question.

3.3 Self-supervised Learning Objective

We aim to **train the MiningNet to learn to specify the most useful text-spans from a given context paragraph for the question.** Therefore, we propose our SSL objective: let the model learn to distinguish important sentences, phrases or words as hints from a noisy context input given a question. For each post, we extract the top rated K(=3) answers and shuffle them randomly to prevent the model remembering to copy the best one always. We concatenate the K answers and feed the resulting text to the unimportant sentences filtering component to form the context. We use the best answer as the (the most useful) answer to the question of the post. Finally we form over 1,900,000 $< question, context, hints >$ triplets for training the model.

3.4 The Implementation and Training Details

HintMiner utilizes the MiningNet to understand the semantics of questions and posts, and then generate suitable answers to the questions. To implement MiningNet, we leverage the transformers library [Wolf et al., 2020]. We use the BERT-base as backbone. For encoder-decoder attention we use 12 attention heads and the hidden size is the same as the BERT. We set hyper-parameters based on previous research[Britz et al., 2017] and the pre-trained BERT encoder structure[Devlin et al., 2018]. Therefore, the hyper-parameters are well fine-tuned.

As the encoder-decoder model suffers from the exposure bias issue [Ranzato et al., 2015, Yuan et al., 2017, He et al., 2016], we adopt a hybrid training strategy which firstly uses teacher forcing training [Ranzato et al., 2015] in a supervised way and then uses the policy gradient reinforcement learning with BLEU4 [Papineni et al., 2002] as reward to fine-tune the model. We leverage the Adam optimizer [Goodfellow et al., 2016] to maximize the probability denoted in Equation 3. We train the model with initial learning rate as $1e-5$ for 200,000 training steps with batch size as 32.

4 Experiments

4.1 Experimental Design

We conducted experiments to evaluate the effectiveness of HintMiner. Our evaluation focuses on the following four research questions:

RQ1: How effective is HintMiner in generating hints?

We compare our model with two kinds of representative methods, i.e. the text retrieval and text generation techniques. We list the compared methods as follows:

- **Text retrieval based methods**. **AnswerBot** [Xu et al., 2017] applies the MMR algorithom [Carbonell and Goldstein, 1998] to the relevant posts given a question to extract proper paragraphs as answers. **SimCSE** is well trained via contrastive learning and can behave rather well in specifying semantic relevant sentences [Gao et al., 2021, Yan et al., 2021]. We retrieve the the sentences with highest cosine similarity for a question from the posts. **PageRank++** extends the traditional graph-based important sentences selection approaches [Erkan and Radev, 2004, Mihalcea and Tarau, 2004] by replacing the tf-idf features of PageRank[Wills, 2006] with semantic vectors of SimCSE.

- **Text generation based methods**. **GPT2** [Radford et al., 2019] an auto-regressive language model that is pre-trained to predict the next to-

Table 1: Evaluation of HintMiner and Compared Methods. "ROU-2" denotes ROUGE-2.

	50		100		150		200	
	BLEU	**ROU-2**	**BLEU**	**ROU-2**	**BLEU**	**ROU-2**	**BLEU**	**ROU-2**
AnswerBot	14.22	16.81	15.63	19.63	17.58	21.27	17.33	21.28
SimCSE	16.53	18.05	16.72	19.62	20.98	22.13	20.11	22.66
PageRank++	19.25	20.02	19.28	22.55	21.84	22.76	19.67	22.84
GPT2	25.11	26.13	26.49	26.83	26.53	28.51	28.04	29.16
BART	29.35	29.57	30.75	31.29	31.52	32.45	33.11	33.17
UniLM	30.00	31.16	33.25	35.17	34.88	33.02	33.69	33.84
HintMiner	32.01	32.28	36.17	36.29	36.09	36.13	34.32	34.67

Table 2: Comparison of HintMiner with Different Relevant Post Retrieval Methods

Retrieval Methods	**BLEU**	**ROUGE-2**
Linked Posts (User Marked)	36.63	36.85
Stack Exchange open API	36.22	36.25
Google Custom Search	36.19	36.37
Elastic Search (HintMiner)	36.17	36.29

ken in text, which is good at many text generation tasks such as summarization, answer generation, etc. **BART** [Raffel et al., 2020] leverages the advantages of both BERT [Devlin et al., 2018] and GPT models [Radford et al., 2019] to pretrain a encoder-decoder based language models by applying several language mask strategies in tokens, sentences and whole documents. **UniLM** [Bao et al., 2020] is a pre-trained unified language model for both auto-encoding and partially autoregressive language modeling tasks using a a pseudo-masked language model, which is good at language understanding and generation.

For each method in baselines, we apply same data processing methods as our HintMiner to prompt fair comparable results. In our implementation, we leveraged the huggingface transformers [Wolf et al., 2020] to build the neural networks.

RQ2: How effective is HintMiner when different relevant post retrieval methods are used?

The relevant post retrieval is an important part of HintMiner. In HintMiner, we use Elastic Search Engine to search for relevant posts. In this RQ, we evaluate the influence of different retrieval methods. In Q&A forums such as Stack Overflow, community members often manually mark the linked posts for some questions. We experimented with the linked posts as the relevant posts (i.e. we directly select posts based on the post-link graph as described for training in Section 2.2.1). Refer to Section 2.2 for more details. We also leverage the open online methods such as Stack Exchange Search Engine API[7] and Google Search Engine API [8] to retrieve three related posts for each test post.

Experimental settings: To evaluate the effectiveness of HintMiner, we randomly sampled 60,000 posts accepted answers that do not appear in our training data. The first paragraph of accepted answer with more than 10 words are **gold hints** of the question. Then, for each question in the posts, we used HintMiner to generate the hints. Finally we used 4-gram BLEU score and 2-gram ROUGE score (ROUGE-2) to evaluate the quality of the generated hints. For RQ1, RQ2 and RQ4, we set the context length to 500 words in our experiments. In the generation decoding process, we use the BEAM-Search algorithm with beam size as 5 and select the best generated texts as the final hint for a question.

4.2 Evaluation Metrics

To evaluate the generated hints, we use two n-gram language model evaluation metrics, i.e. ROUGE and BLEU [Lin, 2004, Papineni et al., 2002], which are widely used in machine translation, summarization, and text generation tasks, etc., to measure the similarity between two sentences. In our research, we measure whether the generated hints are similar to the gold hints. The BLEU score uses the common presence of n-gram count of generated text and reference text to measure the similarity from the precision-like perspective. The ROUGE score measures the similarity from the recall-like perspective. In our experiment, ROUGE uses 2-gram (i.e. ROUGE-2) and BLEU uses 4-gram. Both BLEU and ROUGE-2 scores are 100% when the generated hints are the same as the true hints and 0 when they are totally different. The larger the value, the better the generated hints will be.

[7] `https://api.stackexchange.com`
[8] `https://developers.google.com/custom-search`

Table 3: Examples of Hints Generated by HintMiner

Question	Hints
how do I modify an existing a sheet in an excel workbook using openxlsx package in r? (https://stackoverflow.com/questions/34172353)	you need to load the complete workbook, then modify its data and then save it to disk. with [CODE] you can also specify the starting row and column. and you could also modify other sections.
how do you save android emulator snapshot? (https://stackoverflow.com/questions/4842612/)	start a telnet session to the android emulator then freezes for a few seconds while saving/loading a snaps. i found at google, simply closing the emhot window is the correct way to you.
is there a method to calculate something like general similarity score of a string? (https://stackoverflow.com/questions/4323977/)	there are many such algorithms. keywords are fuzzy string matching. by it you can calculate the number of changes required to transform one string into another, so that gives you an estimate of how similar the strings are.

4.3 Experimental Results for RQ1

The three retrieval baselines are given in Table 1 for the first three rows. It shows the performance results of all the experimented methods for different answer lengths respectively, where $50, 100, 150, 200$ are the answer tokens we truncated. The SimCSE outperforms the AnswerBot (that is based on hand-crafted features) a lot which demonstrate that the capturing the semantics of question and passage through BERT is critical. The BERT++ outperforms the SimCSE, which demonstrates that the importance of sentences in paragraphs concerns a lot and thus it's natural to apply some attention mechanism to better select proper sentences. The HintMiner here directly select text-spans with rather than coarse sentence-level granularity, which significantly outperforms all the baselines.

For the generation based baselines, as shown with middle three rows in Table 1, HintMiner significantly outperforms the three baselines using the proposed MiningNet. The MiningNet obtains a BLEU score of 36.17% and a ROUGE-2 score of 36.29% when the maximum answer length is set to 100 words. The HintMiner outperforms all the baselines given the length of the generated answers varies from 50 to 200 words, which shows the effectiveness of the proposed pretraining objectives. The public models such as GPT2, BART and UniLM is trained using common language modeling objectives, which is not a good solution for the professional situations in our research. Through the unsupervised learning with the huge amount of programming posts, the MiningNet is able to select the needed text spans from the context to generate answers that are semantically similar to the true answers.

We manually checked about 100 questions and hints generated from those methods. The results showed that the sentences that are similar to the question are not necessary the sentences that can form the hints that need to useful for the question rather than just repeat the meaning of question again. Also, the hints are not necessary to be one complete sentence because some contents in paragraphs are not necessary. Therefore, selecting from sentence-level is not rational (i.e. the 3 retrieval baselines). It is also sub-optimal to only consider the common language semantics from Wikipedia or Bookcorpus to pre-train a language model, which lack of knowledge for a specific domains and are not trained to distinguish useful contents as hints in our research problem. In conclusion, these baseline methods cannot effectively generate proper hints given noisy relative posts, which leads to lower performance.

4.4 Experimental Result for RQ2

Table 2 shows the effectiveness of HintMiner when using different methods to find the relevant posts. Using the Linked Posts manually marked by the Q&A community, HintMiner can achieve the best performance, but there are only around 19% of posts marked with links and newly posted questions lack these user marked links. When using Google Custom Search, the search service by Stack Exchange and Elastic Search, both BLEU score and ROUGE-2 score drop a little, but the performance is still acceptable. This experiment also demonstrates the stability and scalability of HintMiner for dealing with different sources retrieved as context passage.

4.5 Examples of the Generated Hints

Table 3 shows some of the hints generated by HintMiner, where the 1^{st} is a solved question (i.e., questions with accepted answers) and the rest are not solved yet. We omit the detailed description of ques-

tions and provide the link to the corresponding Stack Overflow page. The HintMiner can propoerly select text spans of accepted answers against noise paragraphs of sentences (Section 2.2). Although the results may contain grammatical errors they are generally readable and useful. Currently, the generated hints do not contain code or mathematical expression (i.e. represented with symbols such as **[NUM]** and **[CODE]**). For example, the accepted answer of the 2^{nd} question is attached with a code snippet while our generated hints just shows the presence of code here (**[CODE]**).

To further evaluate the effectiveness of HintMiner, we also randomly sampled some without accepted answers. We can see that HintMiner is able to generate meaningful and useful hints even without gold hints in the passage. Taking the 2^{nd} hint as an example, the question is about "usage of simulator" and the answer provides some tips for the question. Those answers further confirm the usefulness of HintMiner. These results are encouraging.

5 Related Work

In recent years, question answering (Q&A) has been receiving a lot of attention in natural language processing.Generally, there are mainly three kinds Q&A systems, including IR based Q&A, KBase based Q&A, and MRC based Q&A. Antonio et al. [Soares and Parreiras, 2018] conducted a literature review in the 130 out of 1842 papers on Q&A systems, and found that 28.57% of the surveyed papers are based on IR and 34.9% on KBase. IR [Croft et al., 2010] is widely studied in Q&A, and combining its with KBases based methods to fetch answers by searching knowledge bases [Dong et al., 2015, Yih et al., 2015] are gaining momentum as some established knowledge bases like FreeBase and DBpedia are publicly available. Recently, many MRC based Q&A methods [Chen et al., 2017, Wang et al., 2018b, Wang et al., 2017] have been proposed. For example, Miller et al. [Miller et al., 2016] used MRC on wikipedia to find the text spans for questions. Currently, these research mainly focus on general open domain Q&A and lack support for the utilization of Q&A forums resources. To help better understand HintMiner, we introduce the MRC and Copy Mechanism here briefly.

5.1 Copying Mechanism

Copying mechanism is inspired by pointer network [Vinyals et al., 2015] that is proposed for OOV problems. It is widely used in many Seq2Seq models [Gu et al., 2016, Zhou et al., 2018]. The copy mechanism selects a set of input tokens to the output. The CopyNet based on the pointer-network [Vinyals et al., 2015, Zhou et al., 2018] directly leverages the attention between the source encoder and the target decoder to generate an output probability distribution over the source tokens.

5.2 Machine Reading Comprehension

Machine reading comprehension (MRC) comprehends a natural language question and then selects a text span (usually not longer than 40 tokens) from a given passage [Rajpurkar et al., 2018, Wang et al., 2018b, Wang et al., 2017, Chen et al., 2017] as the answer to the question. For each question, the task is to select a text span to answer it by outputting a start index and an end index of the input sequence tokens of the passage. For example, DrQA [Chen et al., 2017] select spans over millions of Wikipedia pages to answer general questions such as "who is the current president of USA?". Wang et al. proposed a multi-granularity attention fusion networks [Wang et al., 2018b] to encode the question and the article via multi-granularity attention to select proper text-span. Microsoft researchers also proposed R-Net [Wang et al., 2017] to predict the answer text-span, which leverages the pointer-network for text-span selection.

6 Conclusion

In this paper, we have proposed HintMiner, a machine comprehension and generation based approach to automatic mining hints for users' questions. Given a new question, HintMiner first selects relevant posts and filters away unimportant sentences in the retrieved posts from ES. It then utilizes MiningNet to generate hints to the question from the paragraphs of the relevant posts. MiningNet is an effective self-supervised learning based model, which is able to distinguish proper contents from relevant posts to generate hints. We conduct extensive experiments to evaluate the model effectiveness. The evaluation results show that HintMiner outperforms several important Q&A methods and MiningNet is an effective hints generation neural network. Our tool and experimental data are publicly available `https://github.com/AnonymousAuthor2013/HintMiner`.

In the future, we plan to build a link prediction tool that can better find more relevant posts for a given question. To further improve the capacity of HintMiner, we will also investigate models to comprehend code(**[CODE]**) and numerical expressions (**[NUM]**)in posts. We will also explore more effective metric and perform user studies to evaluate the usefulness of our tool in practice.

References

[Bao et al., 2020] Bao, H., Dong, L., Wei, F., Wang, W., Yang, N., Liu, X., Wang, Y., Gao, J., Piao, S., Zhou, M., et al. (2020). Unilmv2: Pseudo-masked language models for unified language model pre-training. In *International conference on machine learning*, pages 642–652. PMLR.

[Britz et al., 2017] Britz, D., Goldie, A., Luong, T., and Le, Q. (2017). Massive Exploration of Neural Machine Translation Architectures. *ArXiv e-prints*.

[Calefato et al., 2018] Calefato, F., Lanubile, F., and Novielli, N. (2018). How to ask for technical help? evidence-based guidelines for writing questions on stack overflow. *Information and Software Technology*, 94:186–207.

[Carbonell and Goldstein, 1998] Carbonell, J. and Goldstein, J. (1998). The use of mmr, diversity-based reranking for reordering documents and producing summaries. In *Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 335–336. ACM.

[Chen et al., 2018] Chen, C., Chen, X., Sun, J., Xing, Z., and Li, G. (2018). Data-driven proactive policy assurance of post quality in community q&a sites. *Proceedings of the ACM on Human-Computer Interaction*, 2(CSCW):33.

[Chen et al., 2017] Chen, D., Fisch, A., Weston, J., and Bordes, A. (2017). Reading wikipedia to answer open-domain questions. *arXiv preprint arXiv:1704.00051*.

[Croft et al., 2010] Croft, W. B., Metzler, D., and Strohman, T. (2010). *Search engines: Information retrieval in practice*, volume 283. Addison-Wesley Reading.

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

[Dong et al., 2015] Dong, L., Wei, F., Zhou, M., and Xu, K. (2015). Question answering over freebase with multi-column convolutional neural networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, volume 1, pages 260–269.

[Erkan and Radev, 2004] Erkan, G. and Radev, D. R. (2004). Lexrank: Graph-based lexical centrality as salience in text summarization. *Journal of artificial intelligence research*, 22:457–479.

[Gao et al., 2021] Gao, T., Yao, X., and Chen, D. (2021). Simcse: Simple contrastive learning of sentence embeddings. *CoRR*, abs/2104.08821.

[Goodfellow et al., 2016] Goodfellow, I., Bengio, Y., Courville, A., and Bengio, Y. (2016). *Deep learning*, volume 1. MIT press Cambridge.

[Gu et al., 2016] Gu, J., Lu, Z., Li, H., and Li, V. O. (2016). Incorporating copying mechanism in sequence-to-sequence learning. *arXiv preprint arXiv:1603.06393*.

[Haghighi and Vanderwende, 2009] Haghighi, A. and Vanderwende, L. (2009). Exploring content models for multi-document summarization. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 362–370. Association for Computational Linguistics.

[He et al., 2016] He, D., Xia, Y., Qin, T., Wang, L., Yu, N., Liu, T., and Ma, W.-Y. (2016). Dual learning for machine translation. In *Advances in Neural Information Processing Systems*, pages 820–828.

[Koehn and Knowles, 2017] Koehn, P. and Knowles, R. (2017). Six challenges for neural machine translation. In *Proceedings of the First Workshop on Neural Machine Translation, NMT@ACL 2017, Vancouver, Canada, August 4, 2017*, pages 28–39.

[Lin, 2004] Lin, C.-Y. (2004). Rouge: A package for automatic evaluation of summaries. *Text Summarization Branches Out*.

[Mihalcea and Tarau, 2004] Mihalcea, R. and Tarau, P. (2004). Textrank: Bringing order into text. In *Proceedings of the 2004 conference on empirical methods in natural language processing*.

[Miller et al., 2016] Miller, A., Fisch, A., Dodge, J., Karimi, A.-H., Bordes, A., and Weston, J. (2016). Key-value memory networks for directly reading documents. *arXiv preprint arXiv:1606.03126*.

[Papineni et al., 2002] Papineni, K., Roukos, S., Ward, T., and Zhu, W.-J. (2002). Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pages 311–318. Association for Computational Linguistics.

[Radford et al., 2019] Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., Sutskever, I., et al. (2019). Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

[Raffel et al., 2020] Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., and Liu, P. J. (2020). Exploring the limits of transfer learning with a unified text-to-text transformer. *The Journal of Machine Learning Research*, 21(1):5485–5551.

[Rajpurkar et al., 2018] Rajpurkar, P., Jia, R., and Liang, P. (2018). Know what you don't know: Unanswerable questions for squad. *arXiv preprint arXiv:1806.03822*.

[Ranzato et al., 2015] Ranzato, M., Chopra, S., Auli, M., and Zaremba, W. (2015). Sequence level training with recurrent neural networks. *arXiv preprint arXiv:1511.06732*.

[Soares and Parreiras, 2018] Soares, M. A. C. and Parreiras, F. S. (2018). A literature review on question answering techniques, paradigms and systems. *Journal of King Saud University-Computer and Information Sciences*.

[Steinberger and Jezek, 2004] Steinberger, J. and Jezek, K. (2004). Using latent semantic analysis in text summarization and summary evaluation. *Proc. ISIM*, 4:93–100.

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

[Vinyals et al., 2015] Vinyals, O., Fortunato, M., and Jaitly, N. (2015). Pointer networks. In *Advances in Neural Information Processing Systems*, pages 2692–2700.

[Wang et al., 2018a] Wang, S., Chen, T.-H., and Hassan, A. E. (2018a). Understanding the factors for fast answers in technical q&a websites. *Empirical Software Engineering*, 23(3):1552–1593.

[Wang et al., 2018b] Wang, W., Yan, M., and Wu, C. (2018b). Multi-granularity hierarchical attention fusion networks for reading comprehension and question answering. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1705–1714.

[Wang et al., 2017] Wang, W., Yang, N., Wei, F., Chang, B., and Zhou, M. (2017). R-net: Machine reading comprehension with self-matching networks. *Natural Lang. Comput. Group, Microsoft Res. Asia, Beijing, China, Tech. Rep*, 5.

[Wills, 2006] Wills, R. S. (2006). Google's pagerank. *The Mathematical Intelligencer*, 28(4):6–11.

[Wolf et al., 2020] Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., Cistac, P., Rault, T., Louf, R., Funtowicz, M., Davison, J., Shleifer, S., von Platen, P., Ma, C., Jernite, Y., Plu, J., Xu, C., Scao, T. L., Gugger, S., Drame, M., Lhoest, Q., and Rush, A. M. (2020). Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

[Wu et al., 2016] Wu, Y., Schuster, M., Chen, Z., Le, Q. V., Norouzi, M., Macherey, W., Krikun, M., Cao, Y., Gao, Q., Macherey, K., et al. (2016). Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*.

[Xu et al., 2017] Xu, B., Xing, Z., Xia, X., and Lo, D. (2017). Answerbot: automated generation of answer summary to developersź technical questions. In *Proceedings of the 32nd IEEE/ACM International Conference on Automated Software Engineering*, pages 706–716. IEEE Press.

[Xu et al., 2016] Xu, B., Ye, D., Xing, Z., Xia, X., Chen, G., and Li, S. (2016). Predicting semantically linkable knowledge in developer online forums via convolutional neural network. In *2016 31st IEEE/ACM International Conference on Automated Software Engineering (ASE)*, pages 51–62.

[Yan et al., 2021] Yan, Y., Li, R., Wang, S., Zhang, F., Wu, W., and Xu, W. (2021). Consert: A contrastive framework for self-supervised sentence representation transfer. In Zong, C., Xia, F., Li, W., and Navigli, R., editors, *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, ACL/IJCNLP 2021, (Volume 1: Long Papers), Virtual Event, August 1-6, 2021*, pages 5065–5075. Association for Computational Linguistics.

[Ye et al., 2017] Ye, D., Xing, Z., and Kapre, N. (2017). The structure and dynamics of knowledge network in domain-specific q&a sites: a case study of stack overflow. *Empirical Software Engineering*, 22(1):375–406.

[Yih et al., 2015] Yih, S. W.-t., Chang, M.-W., He, X., and Gao, J. (2015). Semantic parsing via staged query graph generation: Question answering with knowledge base.

[Yuan et al., 2017] Yuan, X., Wang, T., Gulcehre, C., Sordoni, A., Bachman, P., Subramanian, S., Zhang,

S., and Trischler, A. (2017). Machine comprehension by text-to-text neural question generation. *arXiv preprint arXiv:1705.02012*.

[Zhou et al., 2018] Zhou, Q., Yang, N., Wei, F., and Zhou, M. (2018). Sequential copying networks. In *Thirty-Second AAAI Conference on Artificial Intelligence*.