

# LiteratureQA: A Question Answering Corpus with Graph Knowledge on Academic Literature

Haiwen Wang, Le Zhou, Weinan Zhang, Xinbing Wang  
Shanghai Jiao Tong University  
Shanghai, China  
{wanghaiwen,zhoule1217,wnzhang,xbwang8}@sjtu.edu.cn

## ABSTRACT

In this paper, we introduce LiteratureQA, a large question answering (QA) corpus consisting of publicly available academic papers. Different from other QA corpus, LiteratureQA has its unique challenges such as how to leverage the structured knowledge of citation networks. We further examine some popular QA method and present a benchmark approach of answering academic questions by combining both semantic text and graph knowledge to improve the prevalent pre-training model. We hope this resource could help research and development of tasks for machine reading over academic text.<sup>1</sup>

## CCS CONCEPTS

• **Information systems** → *Data cleaning*.

## KEYWORDS

Question Answering, Machine Reading Comprehension, Academic Corpus, Academic Knowledge Graph, Academic Network

## ACM Reference Format:

Haiwen Wang, Le Zhou, Weinan Zhang, Xinbing Wang. 2021. LiteratureQA: A Question Answering Corpus with Graph Knowledge on Academic Literature. In *Proceedings of the 30th ACM International Conference on Information and Knowledge Management (CIKM '21), November 1–5, 2021, Virtual Event, QLD, Australia*. ACM, New York, NY, USA, 10 pages. <https://doi.org/10.1145/3459637.3482007>

## 1 INTRODUCTION

With the rapid development of artificial intelligent in recent years, the number of published papers shows an astonishing upward trend (62,300 published in 2018, 84,600 published in 2019 and 92,400 published in 2020)<sup>2</sup>, which requires researchers to spend a large amount of time reading innumerable academic papers for the purpose of research, e.g., for literature survey. Although many paper-reading

groups have been formed to help highlight the most critical information in research papers, time-consuming human efforts are inevitably needed, motivating us to design an intelligent system as an alternative.

Machine reading comprehension (MRC) systems, which enable machines to answer questions about a certain document with a thorough understanding of it, have become an accessible and productive substitute for human labor[7]. Application of MRC to academic papers can save researchers much time on reading papers as well as sorting out papers that meet their requirements.

Many pre-training models such as ELMO [11], BERT [3] and XLNet [26] have achieved great success in several NLP task including MRC and QA. Some researchers use the entity recognising and linking methods to enhance the pre-trained models, such as ERNIE [30]. However, above methods all ignore the graph information among the input academic paper[21], e.g., papers relation in academic social networks for academic paper reading comprehension, which entails the latent graph knowledge and improve the language understanding for tasks such as MRC.

In this paper, we introduce LiteratureQA, the corpus of question answering on academic text, focus on utilizing the enhanced academic paper reading comprehension model with graph information [5] as well as highlighting the essential information in academic paper abstracts in certain topics and mining the latent information in academic social networks. To achieve this goal, we present the paper reading and comprehension dataset (LiteratureQA) which we start with pre-training our model with both textual and structural information and then fine-tuning for the specific question answering task, using the accuracy of answers to represent how machines read and comprehend.

Existing QA datasets, such as SQuAD [12, 13], CNN/Daily Mails [6] and MS MARCO [10] mostly concentrate on general content such as news articles and stories. An alternative and distinctive dataset focusing on academic literature is strongly needed. Inspired by this, we construct a novel QA dataset on academic papers called LiteratureQA, which consists of over 150k question-answer pairs based on a set of over 10k abstracts from artificial-intelligent related papers. These papers are all published on top-tier conferences. The questions are proposed according to the specific context in each abstract, asking about the paper’s objective, method, model, experiment and others. Answers to these questions, consisting of spans, i.e., sequences of words, in the corresponding abstract, are annotated by students with artificial intelligent background. The reasons why LiteratureQA corpus is constrained to research paper abstracts are twofold. First, since an abstract summarizes the main content of a paper, it precisely provides the most concerned information required by researchers, while neglecting trivial details

<sup>1</sup>Our corpus are publicly available for non-commercial use at **Google Drive**: <https://drive.google.com/drive/folders/1Fk978PaB9KGrICMdYHqllgg3ohj-CdhS?usp=sharing>, and our code can be found on <https://github.com/lzhou1998/literatureQA>  
<sup>2</sup>Sources from <https://scholar.google.com>

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from [permissions@acm.org](mailto:permissions@acm.org).  
*CIKM '21, November 1–5, 2021, Virtual Event, QLD, Australia*

© 2021 Association for Computing Machinery.  
ACM ISBN 978-1-4503-8446-9/21/11...\$15.00  
<https://doi.org/10.1145/3459637.3482007>

that one normally shows minor interest in at the very first glimpse of the paper. Second, the long content of academic papers would be very different in structures and presentation of formulas, figures and tables, making it difficult to extract useful and concise information for further use.

Figure 1 shows an example of LiteratureQA corpus. The upper part in light blue includes the metadata information, which is title, author names, institutes, venue, topics, references and abstract. The bottom part in light green consists of 19 questions given to workers, and the answers they made after read the abstract. For the academic entities such as author, institutes, venues, etc., we normalize and distinguish so the same entities are linked by the unified id. For the question “What problem does this paper study?”, the answer is highlighted in the text and presented below as well. Typically, there are several question-answer pairs for a piece of abstract. From this figure, we can see that LiteratureQA has three major characteristics that make it challenging and distinguishing. (1) It is a question answering dataset based on the corpus of academic abstracts. (2) Most of the questions require deep comprehension and reasoning beyond simple word matching or extraction. (3) The answers may contain sophisticated terminology. Such entities require external knowledge to recognize.

To tackle the challenging task and assess the difficulty of LiteratureQA, we benchmark some existing machine comprehension models and propose an intuitive pre-training model, leveraging both text and knowledge graph information to enhance QA performance on our dataset as a benchmark.

Our model mainly consists two modules: (1) the textual encoder (*T-Encoder*), which encode the textual information by capturing the lexical and syntactic information from the input paragraphs or sentences. (2) The graph information encoder (*G-Encoder*), which is designed to integrate the structural information of each paragraph in KG corresponding to the textual input with the underlying layer, so that we can combine the heterogeneous information of both plain text and graph information into the representation. Furthermore, the *T-Encoder* and *G-Encoder* have  $N$  and  $M$  layers respectively.

Empirical results on LiteratureQA corpus show that leveraging knowledge of academic network improves the performance of the prevalent pre-trained language model.

The contributions of our paper are as follows:

- We provide a human-labeled machine reading corpus over 150k question-answer pairs focusing on over 10k academic text, quite different from other existing question answering datasets such as SQuAD [13]. The text in LiteratureQA include metadata such as paper title, authors, normalized institutes, venues and linked references, so every text is build into the academic graph.
- We propose a novel language understanding model as a benchmark method for further comparison by integrating both textual corpora and structural knowledge of academic graph, which outperforms the state-of-the-art pre-training model BERT on the studied tasks.

## 2 RELATED WORK

**Benchmarking Datasets** Reading Comprehension datasets require systems to identify a span in a text to answer a given question,

**Title:** Attention-based Deep Multiple Instance Learning

**Title\_ID:** 344253144

**Authors:** Maximilian Ilse, Jakub M., Max Welling

**Author\_ID:** 1241655402, 1156556698, 1358841117

**Institute:** University of Amsterdam, the Netherlands

**Institute\_ID:** 2101206894

**Topics:** Artificial Neural Network, Interpretability

**Topics\_ID:** 2045245616, 2030024898

**Venue:** International conference on machine learning.

**Venue\_ID:** 2122362464

**References:** Andrews, Stuart, Tsochantaridis, Ioannis, and Hofmann, Thomas. Support vector machines for multiple-instance learning. In NIPS, pp. 577–584 ...

**References\_ID:** 417514186 ...

**Abstract:**

**Multiple instance learning** (MIL) is a variation of supervised learning where a single class label is assigned to a bag of instances. In this paper, we state the MIL problem as learning the Bernoulli distribution of the bag label where the bag label probability is fully parameterized by neural networks. Furthermore, we propose a **neural network-based permutation-invariant aggregation operator** that corresponds to the attention mechanism. Notably, an application of the proposed attention-based operator provides insight into the contribution of each instance to the bag label. We show empirically that our approach achieves comparable performance to the best MIL methods on benchmark MIL datasets and it outperforms other methods on a **MNIST-based MIL dataset and two real-life histopathology datasets** without sacrificing interpretability.

**Question 1:** What problem does this paper study?

**Answer 1:** **Multiple instance learning**

**Question 2:** What approach does this paper propose?

**Answer 2:** **a neural network-based permutation-invariant aggregation operator**

**Question ...**

**Answer ...**

**Question 18:** What dataset does this paper use?

**Answer 18:** **a MNIST-based MIL dataset and two real-life histopathology datasets**

**Figure 1: An illustration of LiteratureQA corpus.**

which typically involves extracting relevant entities and reasoning based on rules. There have been several reading comprehension datasets up to present. MCTest [14] contains 660 stories. Most of the stories and sentences are short, and the size of vocabulary is quite small as well. SQuAD [12, 13], the most famous challenge in the field of question answering, contains about 100K question-answer pairs from 536 articles, where the context for each question is a

single paragraph in these articles. CNN and Daily Mail QA datasets [6] are two large-scale cloze datasets which contain numerous documents. MS Marco [10] is another MRC dataset sampled from real web documents and user queries. Close scrutiny of existing reading comprehension datasets, however, reveals that these datasets do not get involved in the corpus of academic papers. Such corpus presents a more challenging task demanding higher-level intelligence for machines.

At present, there have been several works focusing on the domain of academic literature. The PubMed 200k RCT [2] is a public large-scale dataset for sequential sentence classification built upon academic abstracts. However, this task is simple without requirements for machine comprehension and reasoning. [1] summarizes scientific papers to abstracts and provide two datasets derived from Arxiv and PubMed. However, abstractive summarization is more like information retrieval and lack of comprehension. DL-Paper2Code [15] extracts and understands deep learning design flow diagrams and tables in a research paper and converts them into execution ready source code. SCI TAIL [8], also focusing on scientific QA task, treats multiple-choice question-answering as an entailment problem. It is constructed solely from natural sentences, which is quite different from our reading comprehension dataset. To the best of our knowledge, LiteratureQA is the first dataset bringing machine comprehension into the corpus of academic abstracts.

**Question Answering Models** A great number of end-to-end neural network models have been investigated to tackle the task of machine reading comprehension, including R-Net [23], DCN [25], ReasonNet [16], GA Reader [4] and QANet [27]. They typically consist of an embedding layer, an encoding layer to integrate contextual information, an attention layer to incorporate query and context, a decode layer and an output layer which varies according to the specific QA task. However, in terms of our dataset, empirical observations indicate that word and pattern similarity often misleads the model, contributing to the answer located in a sentence with semantic correlation, which is not supposed to be the location for the right answer. Question understanding and adaption [29] explores different question encoding, but it doesn't adapt to our dataset due to the questions' simple pattern in our dataset. DCR [28] extracts and ranks a set of answer candidates, while we take advantage of semantic information in the sentence level. S-Net [17] and [20] takes advantage of joint learning, inspiring us to design a similar framework [20] for end-to-end training.

### 3 CONSTRUCTING THE CORPUS

In this section, we introduce the process through which we select and clean the publications of the corpus, propose questions and collect answers via crowdsourcing. Then we introduce the way we used to link the entities of the corpus to build the academic graph.

#### 3.1 Document Selection

The recent several years have witnessed an astounding growth in the field of artificial intelligent, especially deep learning in natural language processing and computer vision. The number of publications in this field shows an astonishing upward trend (62,300 published in 2018, 84,600 published in 2019 and 92,400 published

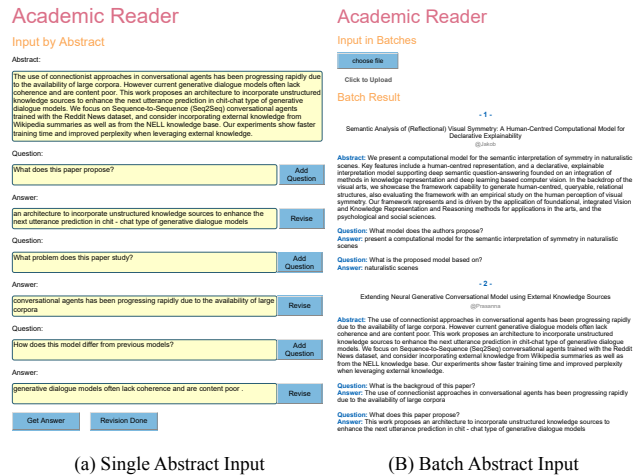


Figure 2: System demonstration for interactive abstract inputs.

in 2020)<sup>3</sup>, which requires researchers to spend a large amount of time reading innumerable academic papers for the purpose of research, so it is very helpful if machine can read papers and answers our questions. Moreover the frontier technologies produced in the field of artificial intelligent are broadly used in other filed such as genetics, geology and etc. The researchers especially in the other field are urgently using the frontier artificial intelligent techniques while it is hard to read so many unfamiliar publications. Therefore it is meaningful establish a brand new question answering corpus focusing on artificial intelligent papers. In addition, papers from different fields vary a lot in structures, contents, objectives, etc. Such distinctions make it hard for machines to learn multifarious patterns. Papers on artificial intelligent share similar patterns, ensuring that the task is learnable. Under these considerations, we sample papers in the field of artificial intelligent published in the last 10 years, mainly sourced from prevalent conferences, including NIPS, ICML, ICLR, AAAI, ACL, and CVPR. Although all of the corpus are related to artificial intelligent, the publications are diverse in terms of topics such as natural language processing, computer vision, neural network theory and etc. Considering the literature in close topic have similar structure of abstract content, and have similar no-answer questions. And the answers to the same question may have similar way of expression. Therefore, it is meaningful for question answering task to consider the topic, keywords and published venues which provide hidden information to a better answer. Moreover, publications written by one author may share some similar expression skills. Leveraging the author information, we can understand the text more easily and precisely and give a better answer. In our corpus, we disambiguate the authors share the same name and normalize the varieties of the author names. For each publication, we use only metadata i.e. title, abstract, authors, institutes of the author, year of publish, topics and references which is no infringement of copyright. We regard title and abstract as pure text for natural language understanding and take author,

<sup>3</sup>Sources from <https://scholar.google.com>

institutes, topics and references as unified entities, so we build the academic graph using entity linking methodology. We collect the open access metadata from academic publisher such as ACM Digital Library<sup>4</sup> and IEEE Xplore<sup>5</sup>, where the metadata can be accessed online by anyone, free of charge and without having to go through any registration.

### 3.2 Question Posing

We establish a question base composed of a finite number of questions, to be exact, 19 different questions as shown in table 1. Questions are rendered through empirical observation of hundreds of abstracts, ensuring that they can be applied to numerous abstracts rather than only a few of them. The questions in the question base are divided into several types of semantic heading as Dernoncourt and Lee’s background, objective, methods, results, conclusions and others which are finite and general. This form of question posing is due to the fact that the more general questions we ask, the deeper comprehension of the abstract is needed. On the contrary, specific and non-unified questions focus on details related to the context, and answers can be retrieved using merely lexical or syntactic variation but not understanding of the academic knowledge. Thus, specific and non-unified questions do not distinguish our datasets from others like SQuAD [12, 13] in that they do not ask about more abstruse academic knowledge, and do not require deeper understanding as well. An example of a question set that we provide according to a specific abstract is shown in Table 1, which is a set of the question base. All of the abstract are asked the same 19 questions, but some of the questions may have no answers.

### 3.3 Answer Sourcing

We create an interactive crowdsourcing website as shown in figure 2, which randomly presents a paper abstract in our corpus following with several questions to the abstract need to be answered. We invite over 500 crowdworkers to assist in building this dataset. Our crowdworkers are college students majoring in computer science who have taken artificial intelligent courses before. Students are awarded bonus according to their performance. Each student provides answers in approximately 20 abstracts on average, and the maximum number of question-answer pairs provided by a single student is about 200. For each The workers are well payed above the local average incomes. Specifically, the workers was payed about 6 dollars for 20 effective reading and answering. Crowdworkers answer questions after acquiring a thorough understanding of the abstract presented. For a given abstract, all the 19 questions are shown on the webpage and the student should read the abstract and answer all of the questions or label as no-answer. They may render the answer null if the abstract contains insufficient information. That is, crowdworkers select questions that they can answer from the prepared question base, thus constructing a specific question set for each abstract as shown in figure 5. Answers can only be attained by highlighting and copying continuous words (i.e. span) from the abstract. We provide our crowdworkers with detailed instructions as well as examples of good and bad answers. The answers selected by crowdworkers can then be stored into our dataset. For each

<sup>4</sup><https://dl.acm.org/>

<sup>5</sup><https://ieeexplore.ieee.org/>

**Table 1: The question set for a specific abstract.**

Category	Question
Objective	What is the objective/aim of this paper?
Objective	What problem(s) does this paper address?
Objective	What are proposed in this paper?
Model	What method/approach does this paper propose?
Model	What is this method based on?
Model	What model does this paper propose?
Model	What is this model based on?
Model	How does the proposed model differ from previous models?
Model	How does the proposed method differ from previous methods/approaches?
Algorithm	What algorithm does this paper propose?
Algorithm	What is this algorithm based on?
Algorithm	How does the proposed algorithm differ from previous algorithms?
Framework	What framework does this paper propose?
Framework	What is this framework based on?
Framework	How does the proposed framework differ from previous frameworks?
Experiment	What experiment does this paper carry out to evaluate the result?
Result	How does this result outperform existing work?
Result	What does the result of this paper show (demonstrated by the experiment)?
Dataset	What dataset does this paper propose?

abstract, it costs about 2 - 10 minutes for workers to read and answer 19 questions. The final clean-up step is done through human efforts as well. To ensure the quality of this dataset as well as evaluate human performance, each answer is scrutinized by at least two crowdworkers. Crowdworkers examine whether the answer is valid, e.g., whether the answer makes sense, serves as an answer to the specific question and is of proper length. Valid answers are maintained. Answers that are entirely nonsense are discarded and re-supplied by our crowdworkers. Moreover, abstracts that has less than five valid answers are discarded.

### 3.4 Graph Construction

For each publications in the LiteratureQA corpus, it contains not only pure text such as title and abstract. Some accessory information are also included such as authors, institutes, published venues, topics and references papers. Intuitively, when we read and comprehend a academic literature, it is helpful if we are familiar to this author and his research interests. We can understand his idea effective and efficient. Moreover, the reference papers may share some common features in the model proposed, the algorithms and the research problem. Therefore, it is essential to leverage the accessory information to understand the academic literature and answer questions. Specifically, there are 5 entities can be used as accessory information, i.e., author, institute, topic, venue and references.

For human readers, we can tell which Micheal Jordan the author is, and what venue such as CIKM is short for The Conference on Information and Knowledge Management. However, the author name disambiguation and variations of entity name normalization is needed for building the corpus. We proposed the novel model leveraging graph knowledge of academic network to improve the performance of machine reading on academic text, while it was ignored by almost all of the prevalent pre-trained language models. In this task, we construct our graph using the method as AceKG [22] dataset. It covers necessary properties of papers, authors, fields of study, venues and institutes, as well as the relations among them. We select several nodes and relations that can reveal the structural information of the whole graph. *Paper nodes* are the basic frame of the graph, and each paper node is related to corresponding *author node*, *venue node*, *field node* as well as its *abstract node*. Selected relations include *paper cite paper*, *paper is published on*, *paper is written by*, *paper is in field* and *field is part of*. Finally, we give each entity a unified ID such as shown in figure 1.

### 3.5 Dataset Analysis

In total, we collect 10,128 abstracts and 153,900 question-answer pairs after final clean-up step. Table 3 counts the number of QA pairs according to question types. Our dataset mainly contains QA pairs focusing on *Objective*, *Method* and *Results*, accounting for 28.5%, 37.4% and 22.1% of the total respectively. *Others* include QA pairs about datasets and experiments. Thus, types of questions are not that unbalanced in number, but they mainly address questions that researchers may develop most interest in. Furthermore, we split our datasets into 2 parts as train/test in an approximate ratio of 6 : 1.

## 4 THE JOINT MODEL

In this section, we present the framework of our model and the detailed implementation. Firstly, we introduce the architecture of our model. Secondly, we elaborate the detail of graph knowledge encoding. Then, we describe the procedures of pre-training and fine-tuning respectively. The code of proposed joint model is part of our resource.

### 4.1 Notations

We denote our constructed corpora knowledge graph as  $G = (V, E)$ , where  $V$  is the node set that includes nodes represent Papers, Authors, Venues, Affiliations and Fields, and  $E \subset V \times V$  is the edge set that stand for their relations. For every input paper abstract  $p_m$ ,  $m = 1, \dots, M$ , we consider the input to be a set of text tokens  $T_m = \{t_{(1,m)}, \dots, t_{(n,m)}\}$ , where  $M$  is the total number of input papers,  $n$  is the length of the each abstract. If the abstract  $p_m$  has a corresponding paper node in  $G$ , then we denote it as  $v_{(m,1)}$  and its one-hop neighbors as  $v_{(2,m)}, \dots, v_{(M_{(m)}+1,m)}$ , where  $M_{(m)}$  is the number of neighbors of  $p_m$ . Because the numbers of neighbors of each paper node are various, so we only choose fixed  $m' - 1$  neighbors for every paper node to keep the model stable, and in this way, we can define the abstract-graph alignment function as  $f(p_m) = \{v_{(1,m)}, \dots, v_{(m',m)}\}$ , and denote  $V_m = \{v_{(1,m)}, \dots, v_{(m',m)}\}$ . We will explain how to select  $m' - 1$  neighbors in the experiment section.

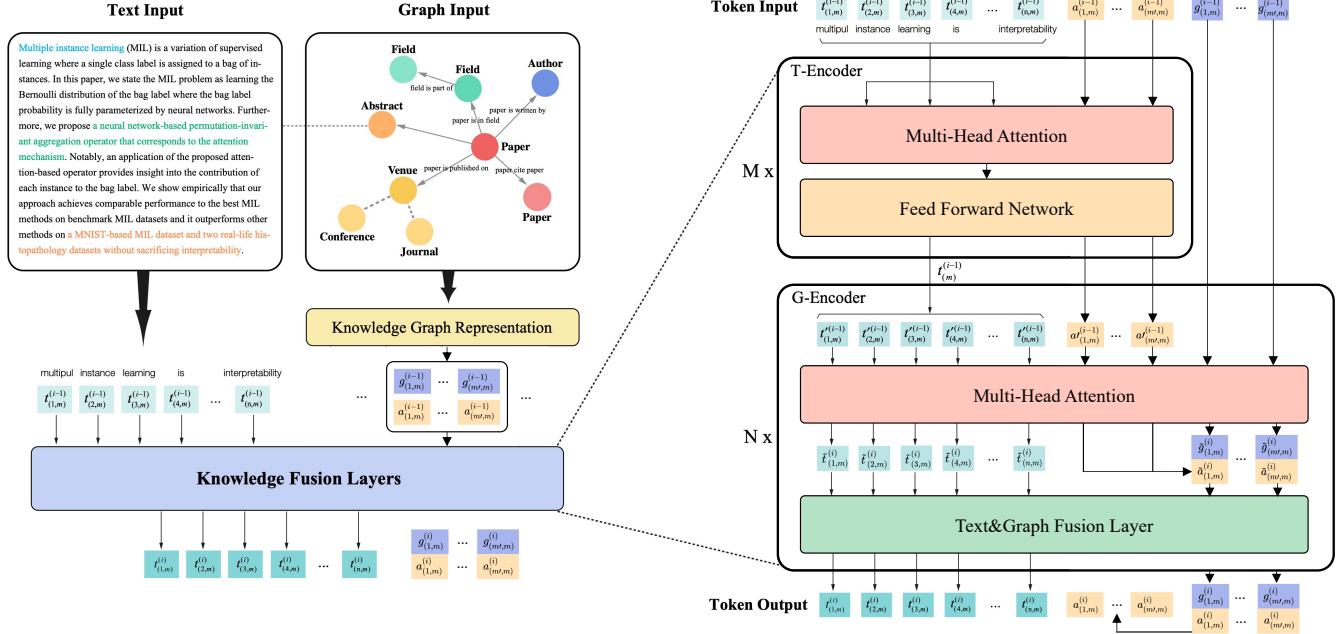


Figure 3: A case study of our system's performance on papers accepted in CVPR.

Considering the abstracts usually not include the text form of entities such as author and venue, so it is hard to directly fuse the information. To solve this problem, we propose anchor tokens to link abstract and those graph entities. For any graph node list  $v = \{v_1, v_2, \dots, v_{n'}\}$ , we introduce the corresponding anchor tokens list  $a = \{a_1, a_2, \dots, a_{n'}\}$ , where the alignment function  $l(v_i) = a_i (i = 1, 2, \dots, n')$ . Due to the number of different nodes in KG is much larger than the number of different tokens, it is impossible to assign a separate anchor token for each entity, so here we let the entities of the same types correspond to the same anchor token. According to our KG, here we introduced 5 anchor tokens corresponding to Paper, Author, Institute, Venue and Field respectively, and randomly initialize their token embeddings. In this way, for an abstract  $p_m$ , the anchor tokens of it can be derived by  $l(f(p_m)) = \{a_{(1,m)}, \dots, a_{(m',m)}\} = A_m$

### 4.2 Model Architecture

The model we proposed consists of two modules: (1) the textual encoder (*T-Encoder*), which encode the textual information by capturing the lexical and syntactic information from the input paragraphs or sentences. (2) The graph information encoder (*G-Encoder*) which is designed to integrate the structural information of each



**Figure 4: The left part is the framework of our model. The right part is the detail of the knowledge fusion layers for integrating the input text and anchor token embeddings and the corresponding graph embeddings. For each iterations, it outputs the tokens embeddings and graph embeddings for the next layer. Mention that  $i$  represent iteration, which means input embeddings through  $i$  layers of T encoder and G encoder. You will find the embeddings from  $(i-1)$ th transit to the  $i$ th on the right part of the figure should actually go through at least 2 iteration, so we use prime symbols to annotate the embeddings in the middle to distinct.**

paragraph in KG corresponding to the textual input to the underlying layer, so that we combine the heterogeneous information of both plain text and graph information into the representation. Furthermore, the *T-Encoder* and *G-Encoder* have  $M$  and  $N$  layers respectively.

Specifically, given an abstract  $p_m$  with its text tokens sequence  $T_m = \{t_{(1,m)}, \dots, t_{(n,m)}\}$ , anchor tokens sequence  $A_m = l(f(p_m))$  and graph nodes sequence  $V_m = f(p_m)$ , the input of the *T-Encoder* is the sum of token embeddings, the segmentation embeddings and the position embeddings for both text and anchor tokens sequence. Utilizing the multi-layer Bi-directional Transformer encoder, we get the lexical and syntactic features  $\mathbf{T}_m = \{\mathbf{t}_{(1,m)}, \dots, \mathbf{t}_{(n,m)}\}$  and  $\mathbf{A}_m = \{\mathbf{a}_{(1,m)}, \dots, \mathbf{a}_{(m',m)}\}$  as follows:

$$\{\mathbf{T}_m + \mathbf{A}_m\} = T\text{-Encoder}(\{\mathbf{T}_m + \mathbf{A}_m\}). \quad (1)$$

where the addition symbols means concatenate.

Considering the module of *T-Encoder* is almost identical to the model proposed in BERT that is so prevalent, we do not describe too much detail information of textual encoder and refer readers in need to Devlin et al., Vaswani et al..

Given the textual embeddings  $\mathbf{T}_m$  and  $\mathbf{A}_m$  from the *T-Encoder*, the *G-Encoder* integrate them with the graph knowledge represent  $\mathbf{G}_m = \{\mathbf{g}_{(1,m)}, \dots, \mathbf{g}_{(m',m)}\}$  that is generated from  $V_m$  by fast and effective graph knowledge embedding model TransE[9]. Then, textual token embeddings  $\mathbf{T}_m$  and  $\mathbf{A}_m$  and graph embeddings  $\mathbf{G}_m$  will be inputted into the *G-Encoder* iteratively to fuse

information from both plain text and the corresponding knowledge graphs before we get the final output  $\mathbf{T}_m^o = \{\mathbf{t}_{(1,m)}^o, \dots, \mathbf{t}_{(n,m)}^o\}$ ,  $\mathbf{A}_m^o = \{\mathbf{a}_{(1,m)}^o, \dots, \mathbf{a}_{(m',m)}^o\}$  and  $\mathbf{G}_m^o = \{\mathbf{g}_{(1,m)}^o, \dots, \mathbf{g}_{(m',m)}^o\}$  for downstream tasks.

$$\{\mathbf{T}_m^o + \mathbf{A}_m^o, \mathbf{G}_m^o\} = G\text{-Encoder}(\{\mathbf{T}_m + \mathbf{A}_m\}, \mathbf{G}_m). \quad (2)$$

### 4.3 Graph-knowledge Encoder

As shown in figure 4, the right part is the module of Graph-knowledge Encoder which is composed of stacked aggregators following the Textual Encoder. For the purpose of integrating both plain text and graph knowledge, the *G-Encoder* fuses the tokens and the graph nodes heterogeneous features simultaneously. For the  $i$ -th iteration, the input text token and anchor token embeddings  $\mathbf{T}_m^{(i-1)}$  and  $\mathbf{A}_m^{(i-1)}$  are firstly fed into a multi-head self attention layer, and graph node embeddings are fed into another one:

$$\{\tilde{\mathbf{T}}_m^{(i)} + \tilde{\mathbf{A}}_m^{(i)}\} = \text{Multi-Head Attention}(\{\mathbf{T}_m^{(i-1)} + \mathbf{A}_m^{(i-1)}\}). \quad (3)$$

$$\{\tilde{\mathbf{G}}_m^{(i)}\} = \text{Multi-Head Attention}(\{\mathbf{G}_m^{(i-1)}\}). \quad (4)$$

Then, the text token and anchor token embeddings after multi-head attention are integrated with the graph embedding  $\tilde{\mathbf{G}}_m^{(i-1)}$  that generated from the last iteration in the information fusion

layer. In this process, for each anchor token embedding  $\tilde{a}_{(j,m)}$  and its aligned graph embedding  $\tilde{g}_{(j,m)}$ , we fuse their information as follow:

$$\begin{aligned} \mathbf{h}_{(j,m)} &= \sigma \left( \tilde{\mathbf{W}}_a^{(i)} \tilde{\mathbf{a}}_{(j,m)}^{(i)} + \tilde{\mathbf{W}}_g^{(i)} \tilde{\mathbf{g}}_{(j,m)}^{(i)} + \tilde{\mathbf{b}}^{(i)} \right), \\ \mathbf{a}_{(j,m)}^{(i)} &= \sigma \left( \mathbf{W}_a^{(i)} \mathbf{h}_{(j,m)} + \mathbf{b}_a^{(i)} \right), \\ \mathbf{g}_{(j,m)}^{(i)} &= \sigma \left( \mathbf{W}_g^{(i)} \mathbf{h}_{(j,m)} + \mathbf{b}_g^{(i)} \right). \end{aligned} \quad (5)$$

where  $1 \leq j \leq m'$ ,  $\mathbf{h}_{(j,m)}$  is the hidden state fusing both the anchor token and the graph information.  $\sigma(\cdot)$  is the non-linear activation function, which is GELU function in our model.

For the text tokens, the information fusion layer computes the output embedding considering only tokens information as follows:

$$\begin{aligned} \mathbf{h}'_{(j',m)} &= \sigma \left( \tilde{\mathbf{W}}_t^{(i)} \tilde{\mathbf{t}}_{(j',m)}^{(i)} + \tilde{\mathbf{b}}'^{(i)} \right), \\ \mathbf{t}_{(j',m)}^{(i)} &= \sigma \left( \mathbf{W}_t^{(i)} \mathbf{h}'_{(j',m)} + \mathbf{b}_t^{(i)} \right). \end{aligned} \quad (6)$$

where  $1 \leq j' \leq m$ .

Simply, we denote the  $i$ -th aggregator operation as follows,

$$\left\{ \mathbf{T}_m^{(i)} + \mathbf{A}_m^{(i)} \right\}, \left\{ \mathbf{G}_m^{(i)} \right\} = \text{Aggregator} \left( \left\{ \tilde{\mathbf{T}}_m^{(i)} + \tilde{\mathbf{T}}_m^{(i)} \right\}, \left\{ \tilde{\mathbf{G}}_m^{(i)} \right\} \right). \quad (7)$$

After the process in the layers of *T-Encoder* and *G-Encoder*, the output embeddings of both tokens and graph computed by the last aggregator would be the final output embeddings of the graph knowledge encoder *G-Encoder*.

#### 4.4 RePre-training

For the purpose of integrate graph knowledge with plain text into language representation, we propose a new pre-training task for our model, which randomly masks some node in the graph that aligned to the anchor tokens and then the system will predict all corresponding nodes based on the abstracts. To do this, we train a denoising auto-encoder similar to Vincent et al. as a node denoising auto-encoder. Considering the size of the whole graph  $G$  corresponding to the corpora is quite large for the softmax layer, the system only predict the node embedding corresponding to the given abstract in stead of all nodes in the knowledge graph. Given the text token embedding sequence  $\mathbf{T}_m^o$ , anchor token embedding sequence  $\mathbf{A}_m^o$  and node embedding sequence  $\mathbf{G}_m^o$  from the encoders, denote  $\mathbf{W}_m = \mathbf{T}_m^o + \mathbf{A}_m^o$ , and we define the aligned node distribution for the token  $w_i$  as follows:

$$p(g_j | w_i) = \frac{\exp \left( \text{linear}(\mathbf{w}_i) \cdot \mathbf{g}_j \right)}{\sum_{k=1}^{m'} \exp \left( \text{linear}(\mathbf{w}_i) \cdot \mathbf{g}_k \right)}, \quad (8)$$

where  $\text{linear}(\cdot)$  is a linear layer. Then, we use Eq. 8 to compute the cross-entropy loss function for the denoising auto-encoder.

Similar to BERT, we adopts both masked language model (MLM) and next sentence prediction (NSP) as pre-training tasks to train learn the lexical and syntactic information from tokens in the input corpora. Anyone in need can also refer BERT to get more detail information. Furthermore, we mask the graph embeddings to capture the graph information for denoising auto-encoder as follows:

(1) We mask 15% of graph nodes aligned to the anchor tokens, in the aim of training our model to learn the graph knowledges. (2) We let 5% of graph nodes align to the wrong anchor tokens, to make the model robust. (2) For the rest of the data, we keep the alignments between tokens and the graph embeddings, which aims to enhance our model to fuse both the graph information and token representations for a better language understanding.

#### 4.5 Fine-tuning

For the downstream tasks, we use the same architectures for both pre-training and fine-tuning except for the output layers. Similar to BERT, the pre-trained model parameters is used to initialize models for downstream tasks. There are straightforward ways to fine-tune the pre-training models by applying the fully-connected layer to the final output embeddings of the given inputs. Therefore, we use the similar method to fine-tune our model, which is inexpensive than pre-training.

### 5 EXPERIMENTS

In this section, we describe the details for graph construction, pre-training and fine-tuning our model on the dataset which contains both plain text and corresponding graph information and benchmark our method on LiteratureQA dataset.

#### 5.1 Parameter Settings and Training Details

In the training process, denoting the hidden dimension of token embeddings and corresponding graph embedding as  $H_w, H_g$  respectively, and the number of self-attention heads as  $A_w$ , we set  $N = 6, M = 6, H_w = 768, H_g = 100, A_w = 12, A_e = 4$ .

Comparatively, the additional knowledge infusion module of the model we proposed has much fewer parameters than BERT, which has about 110M parameters for base model. Therefore, we pre-train our model for only one epoch based on the parameters of well trained BERT model. We deploy most of pre-training hyper-parameters for the textual encoding as same as BERT, except for setting the learning rate as  $3e^{-5}$ . The fine-tuning step shares most of the hyper-parameters for pre-training, except that we set batch size as 16 and training-epochs as 3 respectively.

For neighbor nodes chosen, we set  $m' = 8$ , and the nodes selected are one Paper node itself, one Author node that represents the First Author, one Institute node that represents the First Author's institute, one Venue node that the paper belongs to, and four Field nodes that stand for fields related to the paper. More than 4 fields related to the paper will be reduced to exactly 4 fields according to the correlation between paper and fields from AceKG. For the situation that selected nodes less than 8, we will pad empty nodes with zero vector embeddings.

Because nearly 3000 abstracts and their Q&A pairs are collected and cleaned after our experiment. So, in our experiment, we use part of the final version of LiteratureQA dataset that includes 7000 abstracts with 140000 Q&A pairs, and train/test set is also in an approximate ratio of 6 : 1.

#### 5.2 Evaluation

We provide benchmark methods of several prevalent machine reading models on Academic Reading dataset.

**Table 2: The comparison among RNET, QANet, BERT and our framework on a sample abstract.**

<b>Abstract</b>	We investigate the potential of a restricted Boltzmann Machine (RBM) for discriminative representation learning. By imposing the class information preservation constraints on the hidden layer of the RBM, we propose a Signed Laplacian Restricted Boltzmann Machine (SLRBM) for supervised discriminative representation learning. The model utilizes the label information and preserves the global data locality of data points simultaneously. Experimental results on the benchmark data set show the effectiveness of our method.
<b>Question</b>	What model does this paper propose?
R-NET	a Signed Laplacian Restricted Boltzmann Machine (SLRBM) for supervised discriminative representation learning
QANet	The model utilizes the label information and preserves the global data locality of data points simultaneously.
BERT	a Signed Laplacian Restricted Boltzmann Machine (SLRBM)
<b>Our Model</b>	a Signed Laplacian Restricted Boltzmann Machine (SLRBM)

**Table 3: Number of answered QA pairs according to question types.**

Type	Objective	Method	Results	Others	Total
#(Number)	43,772	57,480	33,922	18,506	153,900

5.2.1 *R-NET*. R-NET [24] is an end-to-end neural network model for question answering task with the formulation of MRC. R-NET first matches the question and the passage with gated attention-based recurrent networks to obtain question-aware passage representation. Then a self-matching attention mechanism is employed to refine the representation by matching the passage against itself. Finally, the pointer networks are applied to locate the positions of answers from the passages. We use the NLPLearn implementation<sup>6</sup>.

5.2.2 *QANet*. QANet[27] is a Q&A architecture which takes first place in SQuAD leaderboard<sup>7</sup>. The encoder of QANet consists exclusively of convolution and self-attention, where convolution models local interactions and self-attention models global interactions. We use NLPLearn implementation<sup>8</sup>. with 740k trainable parameters.

5.2.3 *BERT Fine-tune*. BERT[3] is the prevalent pre-training models which achieved the state-of-the-art results in several NLP tasks. We use pytorch BERT<sup>9</sup> with default parameters to fine-tune on the LiteratureQA.

5.2.4 *Human Performance*. We evaluate human performance on LiteratureQA’s dev and test sets. Recall that answers are scrutinized by at least two crowdworkers during the cleanup stage. We regard answers provided by crowdworkers in the clean-up stage as ground-truth answers, and treat original answers as human predictions.

To evaluate the performance of different models on proposed dataset, we use two metrics. The Exact Match (EM) metric measures the percentage of predictions that match the ground truth answers exactly. The F1 score metric is a less strict metric measuring the overlap between the prediction and the ground truth answers. Both two metrics ignore punctuations and articles(such as *a*, *an* and *the*).

<sup>6</sup><https://github.com/NLPLearn/R-net>

<sup>7</sup><https://rajpurkar.github.io/SQuAD-explorer>

<sup>8</sup><https://github.com/NLPLearn/QANet>

<sup>9</sup><https://github.com/huggingface/pytorch-transformers#Fine-tuning-with-BERT-running-the-examples>

**Table 4: Experiment Results.**

Model	EM	F1
R-NET	11.28	31.69
QANet	18.19	51.26
BERT	66.42	71.28
Ours	<b>67.63</b>	<b>72.75</b>
Human Performance	73.76	86.98

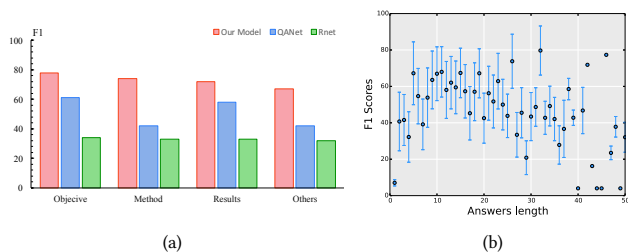
Table 4 illustrates the performance of our model and the aforementioned baseline models on the test set, as well as human performance. Our model achieves EM scores of 67.63 and F1 scores of 72.75 on test set respectively, which beats all the other baseline models. However, there is still a significant gap between our model and human performance. We do notice that EM scores are generally much lower than F1 scores. This is because entities in academic abstracts are often modified by several complex words and clauses, which increases the difficulty of discriminating boundary words and presents a huge challenge of our dataset.

One point of interest is to examine how the performance of our model varies across the lengths of predicted answers. As is shown in Figure 5(a), our model performs stably and well in a wide range of answer lengths. We also note that there is performance degradation when the answer is either too short or too long. This is partly due to the intuitive nature of the evaluation metric.

Moreover, we observe how the performance of each model varies with respect to question types. In Figure 5(b), the height of each bar represents the F1 score. Our model, outperforming the other baseline models in each type except Results, is adept at *Objective* and *Method* questions but struggles with *Results* and *Others* questions. QANet is skilled at *Objective* and *Results* but surprisingly takes a poor performance on *Method*. The performance of R-NET is fairly balanced.

Furthermore, in order to get a further understanding of three models’ robustness and generalization, we collect some latest paper abstracts from Arxiv<sup>10</sup> and run the pre-trained models. Our hands-on evaluation indicates that, typically, all three models can catch the core entities. As the result shown in Table2, all three models

<sup>10</sup><https://arxiv.org>



**Figure 5: Performance of our model across lengths of answers and question types.**

seem to predict the answers with semantic correlation. However, the answer of QANet is only supplementary information introducing the model’s feature and advantage, not the model itself. This indicates that QANet might be fooled by the word *model*. In fact, same as QANet, sequence tagging module of our model also marks the whole sentence as a candidate answer. However, the sentence ranking module gives the previous sentence a higher matching score, leading to dispose of this candidate answer. Both our model and R-NET catch *SLRBM*, the name of the proposed model, while R-NET supplies more details.

### 5.3 Error Analysis

We analyze 50 error examples generated by our model from the test set. We identify four key factors in causing the errors, which are elaborated as below.

**5.3.1 Incorrect Sentence.** In 16% of the examples, the predicted best-matched sentence is incorrect due to semantic-level similarity with the target sentence. For example, as is shown in Table 2, QANet generates a wrong sentence, and our model also makes similar mistakes in some cases.

**5.3.2 Syntactic Complications and Ambiguities.** In 38% of the examples, our model generates sequences containing the same entity as the one that may exist in the correct answer. For instance, for the question “*What method does this paper propose?*”, some spans contain words like approaches and methods etc, which do not exactly refer to the real methods proposed in the papers, but are sometimes marked as answers as well.

**5.3.3 Imprecise Boundaries.** The rest of errors consist of imprecise boundaries where one or more words are either missed or appended at the edge of the correct span. The majority of cases contain an extraneous verb such as propose and present. In other cases, the entire phrase/clause after a conjunction (e.g., *and / or*) is lost. For instance, for the question “*What method does this paper propose?*”, the correct answer is “*a method for object detection and recognition*”.

## 6 CONCLUSION

We introduce LiteratureQA, the largest publicly-available corpus of question answering on academic papers. LiteratureQA consists of over 150k question-answer pairs posed by crowdworkers on a set

of over 10k papers. We aggregate metadata including paper title, authors, topics, abstract, normalized institutes, venues and linked references from multiple sources. Therefore, LiteratureQA can be used effectively for downstream tasks such as constructing academic knowledge graph in academic paper analysis. Moreover, we evaluate several baseline methods including the prevalent pre-trained models, and proposed the state-of-the-art joint model leveraging academic networks to improve the performance as a benchmark approach. We hope LiteratureQA could benefit researchers in automatic paper reading, and the release of our dataset will encourages further exploration.

## ACKNOWLEDGMENTS

This work was supported by National Key R&D Program of China (No.2018YFB2100302), NSF China (No.42050105, 61960206002, 61822206, 62020106005, 61829201), 2021 Tencent AI Lab Rhino-Bird Focused Research Program (No:JR202132) and Shanghai Academic/Technology Research Leader Program (No. 18XD1401800).

## REFERENCES

- [1] Arman Cohan, Franck Dernoncourt, Doo Soon Kim, Trung Bui, Seokhwan Kim, Walter Chang, and Nazli Goharian. 2018. A discourse-aware attention model for abstractive summarization of long documents. *arXiv preprint arXiv:1804.05685* (2018).
- [2] Franck Dernoncourt and Ji Young Lee. 2017. Pubmed 200k rct: a dataset for sequential sentence classification in medical abstracts. *arXiv preprint arXiv:1710.06071* (2017).
- [3] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805* (2018).
- [4] Bhuwan Dhingra, Hanxiao Liu, Zhilin Yang, William W Cohen, and Ruslan Salakhutdinov. 2016. Gated-attention readers for text comprehension. *arXiv preprint arXiv:1606.01549* (2016).
- [5] Aditya Grover and Jure Leskovec. 2016. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 855–864.
- [6] Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *Advances in neural information processing systems*. 1693–1701.
- [7] Yining Hong, Jialu Wang, Yuting Jia, Weinan Zhang, and Xinbing Wang. 2019. Academic Reader: An Interactive Question Answering System on Academic Literatures. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 33. 9855–9856.
- [8] Tushar Khot, Ashish Sabharwal, and Peter Clark. 2018. Scitail: A textual entailment dataset from science question answering. In *Thirty-Second AAAI Conference on Artificial Intelligence*.
- [9] Yankai Lin, Zhiyuan Liu, Maosong Sun, Yang Liu, and Xuan Zhu. 2015. Learning entity and relation embeddings for knowledge graph completion. In *Twenty-ninth AAAI conference on artificial intelligence*.
- [10] Tri Nguyen, Mir Rosenberg, Xia Song, Jianfeng Gao, Saurabh Tiwary, Rangan Majumder, and Li Deng. 2016. MS MARCO: A Human-Generated MACHine Reading COmprehension Dataset. (2016).
- [11] Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365* (2018).
- [12] Pranav Rajpurkar, Robin Jia, and Percy Liang. 2018. Know What You Don't Know: Unanswerable Questions for SQuAD. *arXiv preprint arXiv:1806.03822* (2018).
- [13] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100,000+ questions for machine comprehension of text. *arXiv preprint arXiv:1606.05250* (2016).
- [14] Matthew Richardson, Christopher JC Burges, and Erin Renshaw. 2013. Mctest: A challenge dataset for the open-domain machine comprehension of text. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*. 193–203.
- [15] Akshay Sethi, Anush Sankaran, Naveen Panwar, Shreya Khare, and Senthil Mani. 2018. DLPaper2Code: Auto-generation of code from deep learning research papers. In *Thirty-Second AAAI Conference on Artificial Intelligence*.
- [16] Tao Shen, Tianyi Zhou, Guodong Long, Jing Jiang, Shirui Pan, and Chengqi Zhang. 2018. Disan: Directional self-attention network for rnn/cnn-free language understanding. In *Thirty-Second AAAI Conference on Artificial Intelligence*.
- [17] Chuanqi Tan, Furu Wei, Nan Yang, Bowen Du, Weifeng Lv, and Ming Zhou. 2018. S-net: From answer extraction to answer synthesis for machine reading comprehension. In *Thirty-Second AAAI Conference on Artificial Intelligence*.
- [18] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*. 5998–6008.
- [19] Pascal Vincent, Hugo Larochelle, Isabelle Lajoie, Yoshua Bengio, and Pierre-Antoine Manzagol. 2010. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *Journal of machine learning research* 11, Dec (2010), 3371–3408.
- [20] Di Wang and Eric Nyberg. 2015. A long short-term memory model for answer sentence selection in question answering. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*. 707–712.
- [21] Haiwen Wang, Ruijie Wang, Chuan Wen, Shuhao Li, Yuting Jia, Weinan Zhang, and Xinbing Wang. 2020. Author name disambiguation on heterogeneous information network with adversarial representation learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 34. 238–245.
- [22] Ruijie Wang, Yuchen Yan, Jialu Wang, Yuting Jia, Ye Zhang, Weinan Zhang, and Xinbing Wang. 2018. AceKG: A Large-scale Knowledge Graph for Academic Data Mining. *arXiv e-prints* (2018).
- [23] Shuohang Wang and Jing Jiang. [n.d.]. Machine comprehension using match-LSTM and answer pointer.(2017). In *ICLR 2017: International Conference on Learning Representations, Toulon, France, April 24-26: Proceedings*. 1–15.
- [24] Wenhui Wang, Nan Yang, Furu Wei, Baobao Chang, and Ming Zhou. 2017. Gated self-matching networks for reading comprehension and question answering. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 189–198.
- [25] Caiming Xiong, Victor Zhong, and Richard Socher. 2016. Dynamic coattention networks for question answering. *arXiv preprint arXiv:1611.01604* (2016).
- [26] Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Ruslan Salakhutdinov, and Quoc V Le. 2019. XLNet: Generalized Autoregressive Pretraining for Language Understanding. *arXiv preprint arXiv:1906.08237* (2019).
- [27] Adams Wei Yu, David Dohan, Minh-Thang Luong, Rui Zhao, Kai Chen, Mohamad Norouzi, and Quoc V Le. 2018. Qanet: Combining local convolution with global self-attention for reading comprehension. *arXiv preprint arXiv:1804.09541* (2018).
- [28] Yang Yu, Wei Zhang, Kazi Hasan, Mo Yu, Bing Xiang, and Bowen Zhou. 2016. End-to-end answer chunk extraction and ranking for reading comprehension. *arXiv preprint arXiv:1610.09996* (2016).
- [29] Junbei Zhang, Xiaodan Zhu, Qian Chen, Lirong Dai, Si Wei, and Hui Jiang. 2017. Exploring question understanding and adaptation in neural-network-based question answering. *arXiv preprint arXiv:1703.04617* (2017).
- [30] Zhengyan Zhang, Xu Han, Zhiyuan Liu, Xin Jiang, Maosong Sun, and Qun Liu. 2019. ERNIE: Enhanced Language Representation with Informative Entities. *arXiv preprint arXiv:1905.07129* (2019).