Received November 29, 2020, accepted December 23, 2020, date of publication December 30, 2020,
date of current version January 8, 2021.

Digital Object Identifier 10.1109/ACCESS.2020.3048164

Enhanced Natural Language Interface for Web-Based Information Retrieval

TIAN BAI[1,3], (Member, IEEE), YAN GE[2,3], SHUYU GUO[1,3], ZHENTING ZHANG[1,3], AND LEIGUANG GONG[4]
[1]College of Computer Science and Technology, Jilin University, Changchun 130012, China
[2]College of Software, Jilin University, Changchun 130012, China
[3]Key Laboratory of Symbolic Computation and Knowledge Engineering, Ministry of Education, Jilin University, Changchun 130012, China
[4]Yantai Huashen Intelligent Technology Limited, Yantai 264000, China

Corresponding author: Tian Bai (baitian@jlu.edu.cn)

This work was supported in part by the National Natural Science Foundation of China under Grant 61702214, in part by the Development Project of Jilin Province of China under Grant 20200801033GH and Grant 2020122328JC, in part by the Jilin Provincial Key Laboratory of Big Data Intelligent Computing under Grant 20180622002JC, and in part by the Fundamental Research Funds for the Central University, Jilin University.

ABSTRACT Database application is at the core of most web application systems such as web-based email, source codes repository management, public scientific data repository management, news portals, and publication repository of various fields. However, the usage of these database systems for data and information retrieval is severely limited because of lacking support for processing search queries expressed in a natural language (NL). Most web interfaces for databases today only take search queries entered in some form of logical combination of keywords or text strings, which restrict the scope and depth of what a web user really wants to search for, even though natural language based data or information retrieval has made significant advances in recent years. To overcome or at least to alleviate such limitation in web information services, we propose in this article an improved neural model based on an existing framework IRNet for NL query of databases, in which a representation of Gated Graph Neural Network (GGNN) is introduced to encode the database entities and relations. We also represent and use the database values in the prediction model to identify and match table and column names for automatic synthesize a correct SQL statement from a query expressed in a NL sentence. Experiments with a public dataset demonstrates the promising potential of our approach.

INDEX TERMS Neural network, natural language processing, text-to-SQL, gated graph neural network.

I. INTRODUCTION

Nowadays database (DB) application is the backbone of most web-based information services such as web-based email, source codes repository management, public scientific data repository management, news portals, and publication repositories of various fields [1]–[3]. Fig. 1 is a snapshot of a website Web of Science [4] that provides an interface for searching scientific publications from its database of scientific citations. Like most similar web database interfaces [5]–[7], it supports search method of using keywords as well as complex formulas with identifier, Boolean operators, and brackets. Such search method is more advanced and powerful than simple keywords-based search [8]–[10]. But its actual search power may be limited by its complexity which requires some level of knowledge about the database content and the expertise of using the search tool. To most users without such knowledge and expertise, most likely they will not be able to take full advantage of the search tool for their data or information needs. Such limitation can only be overcome or at least alleviated by a natural language interface with the support of NL query to SQL query (NL-SQL) or text to SQL (TTS) capabilities. Please note ''NL-SQL'' and ''TTS'' will be used interchangeably in this article.

In research of TTS, deep learning (DL) has been the primary choice for the last few years. Many variations of DL algorithms have been developed and tested using WikiSQL dataset [11] a first large-scale Text-to-SQL (TTS) dataset. Zhong *et al.* [11] proposed Seq2SQL, a model based on Seq2Seq structure and utilizing reinforcement learning to generate SQL queries; Xu *et al.* [12] proposed SQLNet based on Seq2SQL, which uses a sequence-to-set model and a col-

The associate editor coordinating the review of this manuscript and approving it for publication was Long Wang.

umn attention mechanism to substantially improve accuracy without the use of reinforcement learning; Yu *et al.* [13] proposed TypeSQL, which views the TTS problem as a slot filling task and uses type information to better understand rare entities and numbers in the natural language questions; Dong *et al.* [14] proposed a structure-aware neural network that decomposes the semantic parsing process into two stages: first, generating its sketch based on the input sentences, and then, filling in missing details by considering natural language and the sketch itself, later, with the advent of more and better pre-trained language models, increasing number of researchers applied them to improve their TTS algorithms on WikiSQL, such as SQLova [15] which incorporates the BERT model [16].

WikiSQL dataset is limited for it only contains databases of single table. Yu *et al.* [17] proposed a large-scale, complex, and cross-domain Text-to-SQL dataset Spider containing databases of multiple tables. Spider dataset supports investigations of sophisticated NLP and DL models for predicting a large number of complex SQL queries. Some recent studies using Spider have developed more advanced methods and algorithms achieving impressive results. SyntaxSQLNet [18] is the first model developed for the Spider task using a syntax tree representing the features of the SQL queries. It also proposed a method for generating cross-domain training data to enhance model performance with data augmentation. Lee [19] proposed a SQL clause-wise decoding neural architecture with a self-attention based database schema encoder for the Spider task. Wang *et al.* [20] presented a unified framework, called RAT-SQL, based on the relation-aware self-attention mechanism, to address schema encoding, schema linking, and feature representation within a text-to-SQL encoder. Bogin *et al.* [21] presented an encoder-decoder semantic parser, where the structure of the DB schema is encoded with a graph neural network (GNN) [22].

Guo *et al.* [23] propose a very interesting deep neural network based approach IRNet to tackle complex and cross-domain Text-to-SQL problems using Spider dataset. By decomposing an TTS tasks into three phases and using an intermediate representation [24]–[28], IRNet not only provides an effective alternative approach to addressing the mismatch problems and difficulties of predicting columns caused by the large number of out-of-domain words, but also presents ''break-point'' for intermediate performance analysis which is what really interests us most. For instance, in IRNet an intermediate representation called SemQL is designed to bridge NL and SQL. An SQL query is inferred from the synthesized SemQL of the query with domain knowledge. Such capability allows us to perform certain analyses of intermediate results to gain better understanding of parts of its DL algorithm, so that targeted revision or algorithm improvement can be made. Because the rich expressiveness of this representation, we will adopt it in our extended implementation of IRNet [23].

In analyzing IRNet performance on Spider dataset, we found database values, as an important part of the database, can provide valuable information for database field prediction. In this article we will introduce, in the framework of IRNet, database value into the prediction model to identify and match table and column names in natural language queries.

To investigate and understand what we can do to further improve the performance of IRNet, we have conducted a series of experiments, and performed some in-depth analyses of causes of mismatches. We identify two causes: (1) mismatches due to lack of the representations of relations between tables and (2) mismatches due to lack of the representations of database values. To address these two issues, we propose two extensions to the IRNet: (1) add Gated Graph Neural Network (GGNN) [21], [22] to IRNet to encode the database structure; (2) represent database values in the prediction model.

II. METHODS

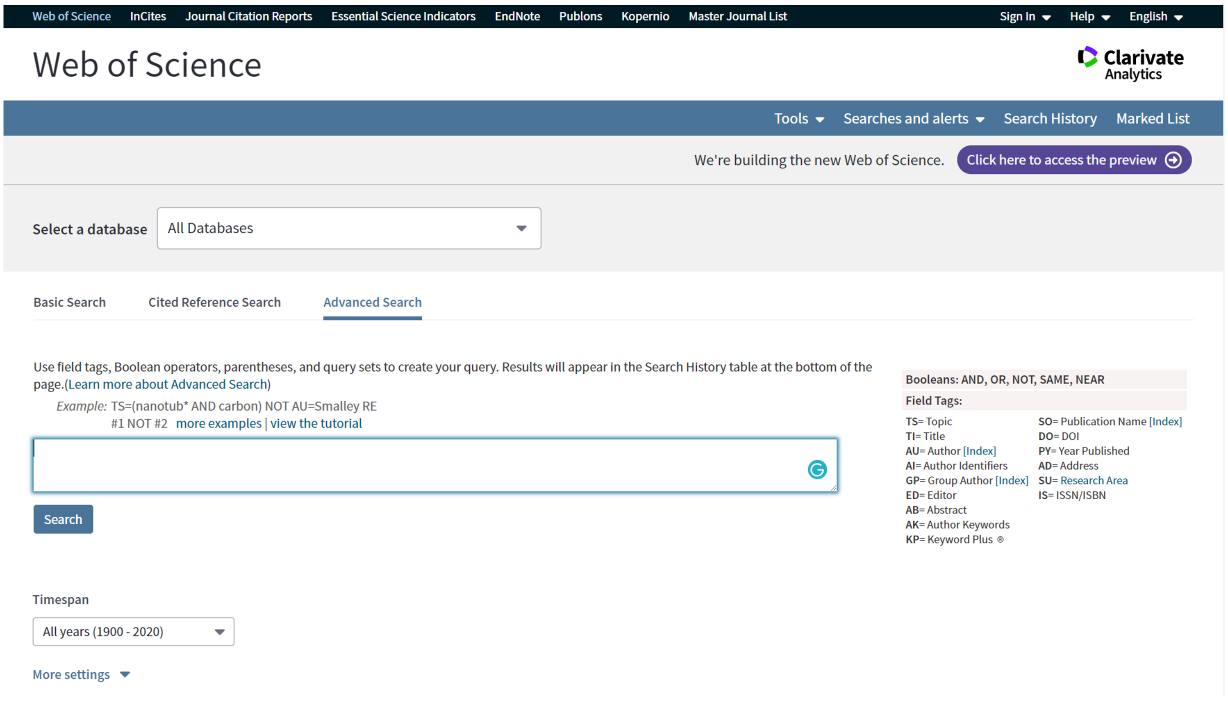
Database values can often provide valuable information or clues about the correspondences between words in a NL query and database fields, which can be used to improve the accuracy of identification of targeted table and column. Schema of database structures i.e. tables, columns and their relations are informatively invaluable for predicting SQL queries. How to represent and use such information in a deep neural network-based model has profound impact to its performance.

A. BASE MODEL

We adopt basic IRNet [23] framework (Fig. 2(a)) in our implementation, an overview of which is shown in (Fig. 2(b)). An TTS task is carried out within the framework in three phases or subtasks. In the first phase, words or tokens in an NL query are paired with database fields by Schema Linking. Subsequently, target columns and tables are determined by leveraging database values together with database schema in the prediction model. We introduced a representation of Gated Graph Neural Network (GGNN) [21], [22] to encode the DB schema replacing the original IRNet representation of DB schema. Relations between DB tables and relations between tables and columns missing in IRNet are now fully represented and used in the model. An attention mechanism [29] is implemented to compute the Value Attention Embedding to be added in the column Embedding, which helps the model to identify potential table and column names in the NL query. An intermediate representation (IR) for the query is obtained and represented in a domain-specific language, called SemQL [23]. A syntax-based neural network model is used to synthesize SemQL query. Finally, A SQL query is generated based on SemQL and domain knowledge [23]. Transforming SemQL to SQL is done by traversing the SemQL tree from its root to leaf nodes.

B. EXTRACTING TABLE DATA FROM WEB PAGES

To obtain data on a web page and organize it in a formalized way using database, we analyze the tags corresponding to the



Web of Science | InCites | Journal Citation Reports | Essential Science Indicators | EndNote | Publons | Kopernio | Master Journal List | Sign In | Help | English

Web of Science

Clarivate Analytics

Tools | Searches and alerts | Search History | Marked List

We're building the new Web of Science. Click here to access the preview

Select a database: All Databases

Basic Search | Cited Reference Search | Advanced Search

Use field tags, Boolean operators, parentheses, and query sets to create your query. Results will appear in the Search History table at the bottom of the page.(Learn more about Advanced Search)

Example: TS=(nanotub* AND carbon) NOT AU=Smalley RE
#1 NOT #2 more examples | view the tutorial

Search

Timespan
All years (1900 - 2020)

More settings

Booleans: AND, OR, NOT, SAME, NEAR

Field Tags:

TS= Topic	**SO**= Publication Name [Index]
TI= Title	**DO**= DOI
AU= Author [Index]	**PY**= Year Published
AI= Author Identifiers	**AD**= Address
GP= Group Author [Index]	**SU**= Research Area
ED= Editor	**IS**= ISSN/ISBN
AB= Abstract	
AK= Author Keywords	
KP= Keyword Plus ®	

FIGURE 1. **Web of science's query page, when using advanced search, users are required to create queries according to its grammar.**

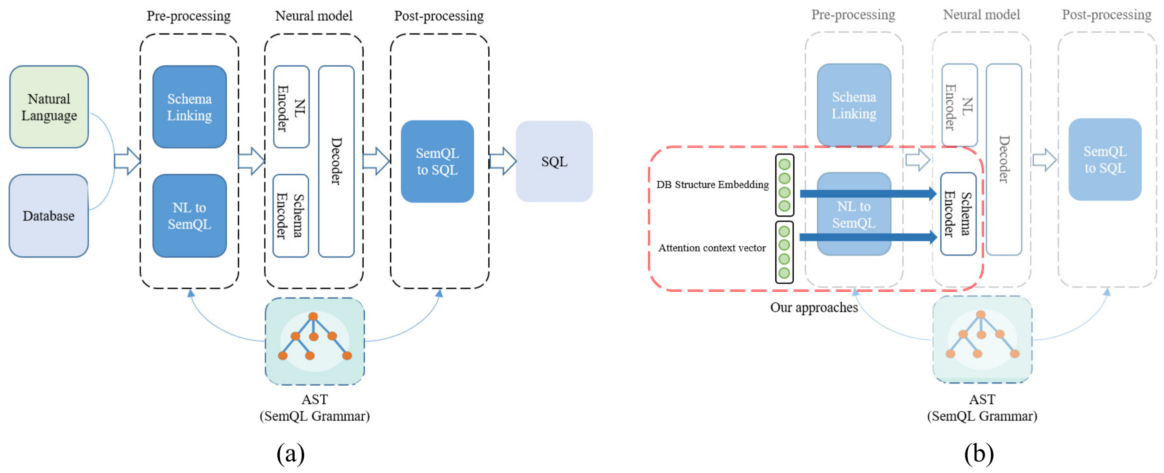
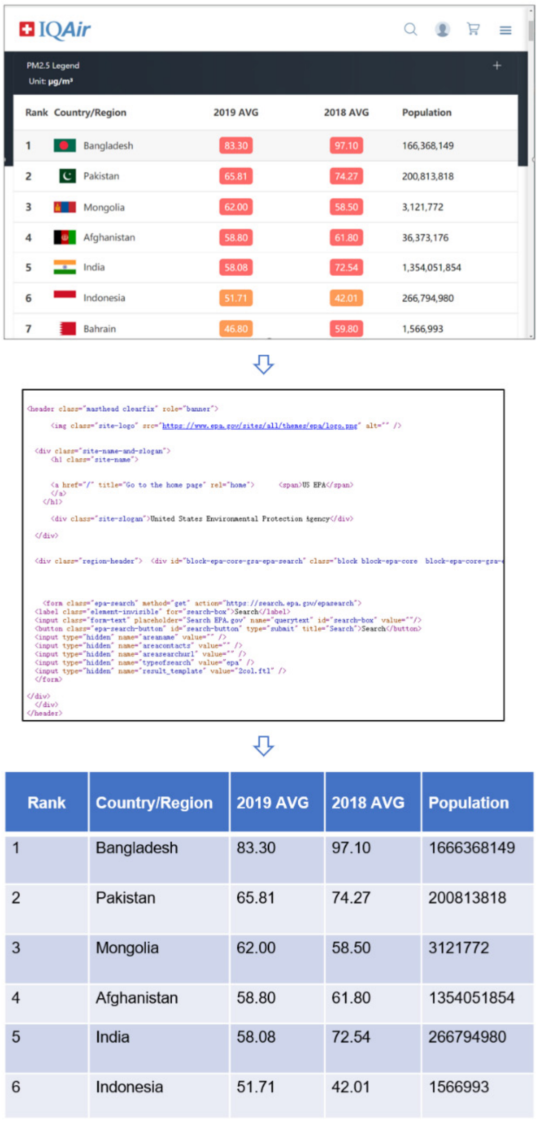


FIGURE 2. **The IRNet model and our improvements based on it. (a) is an overview of the IRNet model. The three subtasks of IRNet are marked with a black dashed line. In the preprocessing stage, Schema Linking identifies the table and column in the NL, and the NLwill be converted to SemQL according to the IR grammar. The SemQL output by the neural network is then inferred into SQL statements according to the grammar in the post-processing stage. (b) is an illustration of how our methods work (Marked by a red dashed line). We calculate two Embeddings through DB Schema and DB Values, they will be used in IRNet's Schema encoder.**

table in the HTML code of the webpage. In general, web pages contain a lot of information. Each web page may be composed of text, tables, pictures, links, etc. Besides these, there are also a large number of HTML tags and CSS [30] styles that are used to control the layout and display of the web. The target data that we really need to extract is scattered among the above-mentioned different forms of HTML components. Therefore, it is necessary to locate the tables from the web page [31], [32].

After locating the table on the webpage, we must further identify the validity of the table [33], [34]. In addition to displaying data and information in tabular form, tables in web pages can also be used to generate layouts and show effects. Because of the multiple uses of HTML table tags, the table area usually contains some invalid information such as web page layout and advertisement. Thus, before extracting table data, invalid tables must be removed to keep the data that is really needed.

We use a web crawler [35], [36] to get the table data. The general crawler routine is nothing more than the steps of sending a request, obtaining a response, parsing a web page, extracting data, and saving data. For a complete crawler,



IQAir

PM2.5 Legend
Unit: μg/m³

Rank	Country/Region	2019 AVG	2018 AVG	Population
1	Bangladesh	83.30	97.10	166,368,149
2	Pakistan	65.81	74.27	200,813,818
3	Mongolia	62.00	58.50	3,121,772
4	Afghanistan	58.80	61.80	36,373,176
5	India	58.08	72.54	1,354,051,854
6	Indonesia	51.71	42.01	266,794,980
7	Bahrain	46.80	59.80	1,566,993

```
<header class="masthead clearfix" role="banner">

      <img class="site-logo" src="https://www.epa.gov/sites/all/themes/epa/logo.png" alt="" />


  <div class="site-name-and-slogan">
      <h1 class="site-name">


      <a href="/" title="Go to the home page" rel="home">       <span>US EPA</span>
      </a>
    </h1>

      <div class="site-slogan">United States Environmental Protection Agency</div>

  </div>


  <div class="region-header">  <div id="block-epa-core-gsa-epa-search" class="block block-epa-core  block-epa-core-gsa-e



    <form class="epa-search" method="get" action="https://search.epa.gov/epasearch">
  <label class="element-invisible" for="search-box">Search</label>
  <input class="form-text" placeholder="Search EPA.gov" name="querytext" id="search-box" value=""/>
  <button class="epa-search-button" id="search-button" type="submit" title="Search">Search</button>
  <input type="hidden" name="areaname" value="" />
  <input type="hidden" name="areacontacts" value="" />
  <input type="hidden" name="areasearchurl" value="" />
  <input type="hidden" name="typeofsearch" value="epa" />
  <input type="hidden" name="result_template" value="2col.ftl" />
  </form>

</div>
  </div>
</header>
```

Rank	Country/Region	2019 AVG	2018 AVG	Population
1	Bangladesh	83.30	97.10	1666368149
2	Pakistan	65.81	74.27	200813818
3	Mongolia	62.00	58.50	3121772
4	Afghanistan	58.80	61.80	1354051854
5	India	58.08	72.54	266794980
6	Indonesia	51.71	42.01	1566993

FIGURE 3. **The process of obtaining tables on the web. We use python tools to extract tables from HTML source code.**

the amount of code can range from dozens of lines to hundreds of lines, and the cost is relatively high, so we take the read_html function of the Pandas tool [37] in Python as our approach. As shown in Fig. 3, first, we send a request to the specified website URL and get the source code of the page, then, for the purpose of locating the table in the HTML page and filter out the redundant information, we apply XPath combined with manual features to preprocess. After that, we use read_html function in Pandas to parse the potential data, this function can directly capture the table in the web page. Finally, we store the parsed tabular data in a csv file and import the csv file into the SQL database.

C. COLUMN REPRESENTATION COMBINED WITH DB VALUE

To compute possible correspondences between words in an NL query and database values, we enumerate all n-grams [37] of length 1-6 in a NL query sentence, and then compute the similarity of these n-grams to the database values:

$$sim\,(words, content) = DLdistance(word, content) \quad (1)$$

where *words* in (1) denotes the n-grams, and *content* denotes the database value. If the similarity exceeds a certain threshold, the corresponding database value is taken as selected.

An attention mechanism [38] (Fig. 4) is then applied to the value and the corresponding natural language question, so that the NL query sentence can carry potentially valuable information of the database value for column prediction.

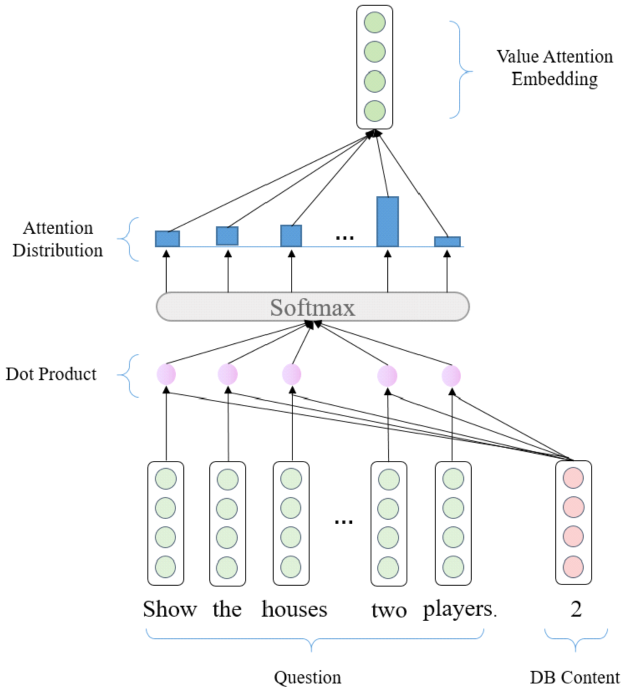


FIGURE 4. **Calculation of Value attention embedding, In this process we employ the attention mechanism.**

$$E_{cont} = \begin{cases} Embedding(content) & sim\,(words, content) > \lambda \\ 0 & else \end{cases} \quad (2)$$

$$w_v = softmax\left(E_v^T W_v E_q\right) \quad (3)$$

$$E_{q|v} = w_v E_q \quad (4)$$

where *Embedding* in (2) is a function that converts a string to a corresponding embedded matrix, λ is the similarity threshold. E_v in (3) is the Embedding of the selected database value, E_q is the Embedding of the natural language question, W_v is the parameter matrix, w_v represents the attentional weight of the database value on the natural language question, and $E_{q|v}$ in (4) is the Embedding of the natural language question that combines the database value.

Finally, the $E_{q|v}$ is added to the IRNet column Representation (column Embedding) as a Value Attention Embedding.

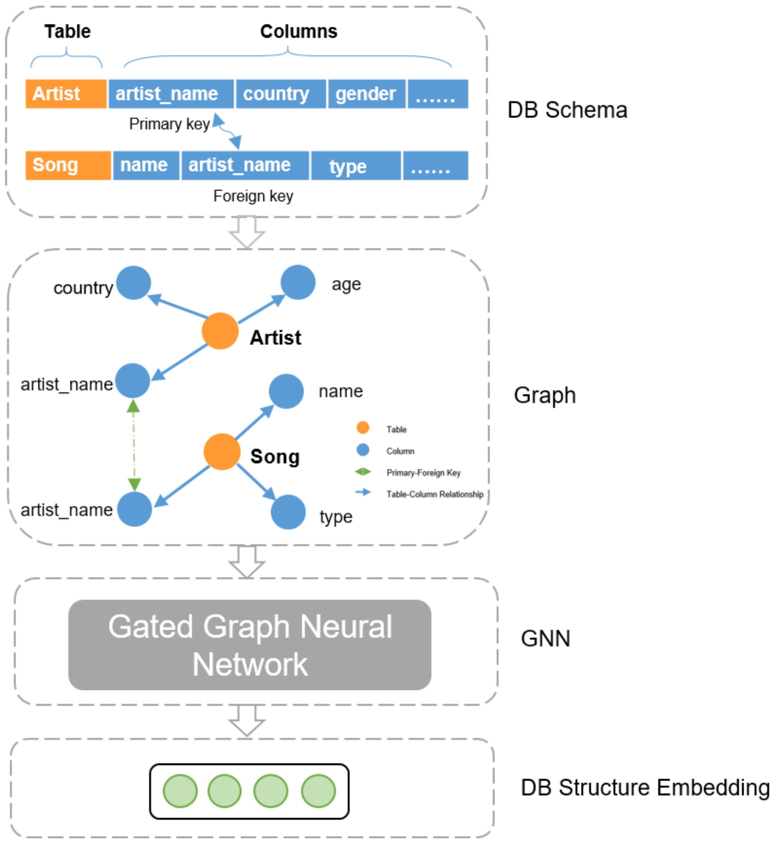


FIGURE 5. **Encode DB Schema with a GGNN. First, the DB Schema is represented as a graph, and then input into GGNN, finally, the output embedding of the DB Schema containing global information is obtained.**

D. ENCODING DB SCHEMA WITH GRAPH NEURAL NETWORK

As shown in Fig. 5, Gated Graph Neural Network (GGNN) [21], [22] is used in IRNet to encode database structure, where tables and columns are represented as nodes. Affiliation relationships between columns and tables as well as the primary foreign key relationship are represented as edges. In order to include more information about the relationships between tables and columns, column Embedding and table Embedding are computed from the GGNN instead of the initial embedding in IRNet's Schema Encoder.

$$h_v^{(0)} = x||0 \tag{5}$$

$$a_v^{(l)} = \sum\nolimits_{type \in \{\rightarrow, \leftrightarrow\}} \sum\nolimits_{(u,v)\in \varepsilon_{type}} W_{type} h_u^{l-1} + b_{type} \tag{6}$$

$$h_v^{(l)} = GRU(h_v^{(l-1)}, a_v^{(l)}) \tag{7}$$

The x in (5) denotes node features, which is filled with 0 if there are not enough dimensions, and $h_v^{(0)}$ is used as the initial state of the node. The $\rightarrow$, $\leftrightarrow$ in (6) denotes the two types of edges, and ε_{type} is the set of edges, in which each node recalculates its representation in each step according to the representation of its neighbors in the previous step. (7) denotes the final representation of each node computed by GRU [39]. In the GGNN approach, each node represents a table or a column, and the final representation contains a global schema structure.

The calculation process is as follows:

1. generate the edge vector of the graph network according to the relationship between table and column, there are two kinds of edge vectors:
   1) The affiliation between table and column (which table the column belongs to)
   2) Primary Foreign Key Relationships between columns
2. Merge the table and column vectors into a single node vector
3. Apply the GNN recursively and get the DB Structure Embedding. At each step, each node re-computes its representation based on the representation of its neighbors in the previous step.

III. EXPERIMENTS

Since the SQL used when querying the web-based information is relatively simple, it may not contain complex SQL components, such as Join, Group by, Union etc., so we extracted some simple samples from the Spider data set that match the difficulty of the web query to test our model performance. The purpose of our experiments is to demonstrate

TABLE 1. Parameter Settings of the Model.

Hidden vectors	Action embedding	Node-type embedding	dropout rate	Batch size	GGNN iteration
300	128	64	0.3	64	3

how much performance improvement can be gained by our proposed extensions to the IRNet and its ability to query the web-based information. As such the experiments were designed to only test the original IRNet implementation and our proposed extended implementation, based on which a preliminary comparative evaluation will be made.

A. DATASET

In our experiments, Spider [17] dataset is used, which contains over 10,000 natural language questions and their corresponding SQL statements from 200 databases of 138 domains with multiple tables, each database contains 5.1 tables on average. In the Spider data set, each query sample has a label, and the difficulty level is determined according to the number of SQL components, selections, and conditions. There are 4 levels of difficulty, for example, a SQL query containing only SELECT, FROM, and WHERE components will be marked as simple. And if a query contains more than two SELECT columns, two or more WHERE conditions and GROUP BY two columns, or contains EXCEPT or nested queries, it is considered difficult.

B. IMPLEMENTATION

We implement our TTS system with PyTorch [40]. Following IRNet, parameters for the neural network are chosen empirically as shown in Table 1. The dimension of hidden vector is set to 300. The dimensions of action embedding and node type are 128 and 64. Word embedding is initialized using Glove [41] and shared between the NL encoder and the Schema encoder. Adam [42] is selected as the optimizer, and the hyper parameters are all default settings. Dropout rate [43] and Batch size are set to 0.3 and 64 respectively. The iterations of GGNN is 3.

C. RESULTS

When the user expresses a natural language query for web-based information, it is often not too complicated, they tend to ask a simple question like "How many acting statuses are there?" rather than a more sophisticated one "Show the name and number of employees for the departments managed by heads whose temporary acting value is 'Yes'?". In general, SQL statements corresponding to complex natural language queries are also more difficult to parse. To adapt to the features of retrieving web-based information, we simplified the Spider data set to make the samples more similar to those when retrieving network information to verify the performance of the model during this work. Table 2 shows the model's performance on samples similar to those often appear in web-based information retrieval.

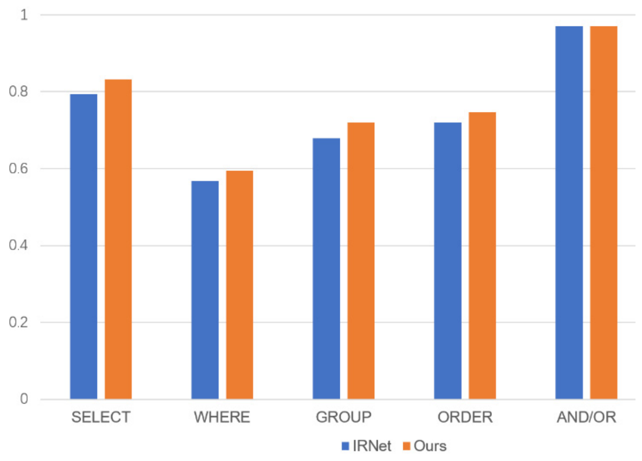


FIGURE 6. F1 scores of component matching of IRNet and Our methods.

TABLE 2. Exact Match Accuracy on the Samples Similar to Web-Based Information.

	IRNet	Ours
Exact Match Accuracy	67.3%	70.5%

TABLE 3. Overall Results and the Performance of Each Difficulty Level.

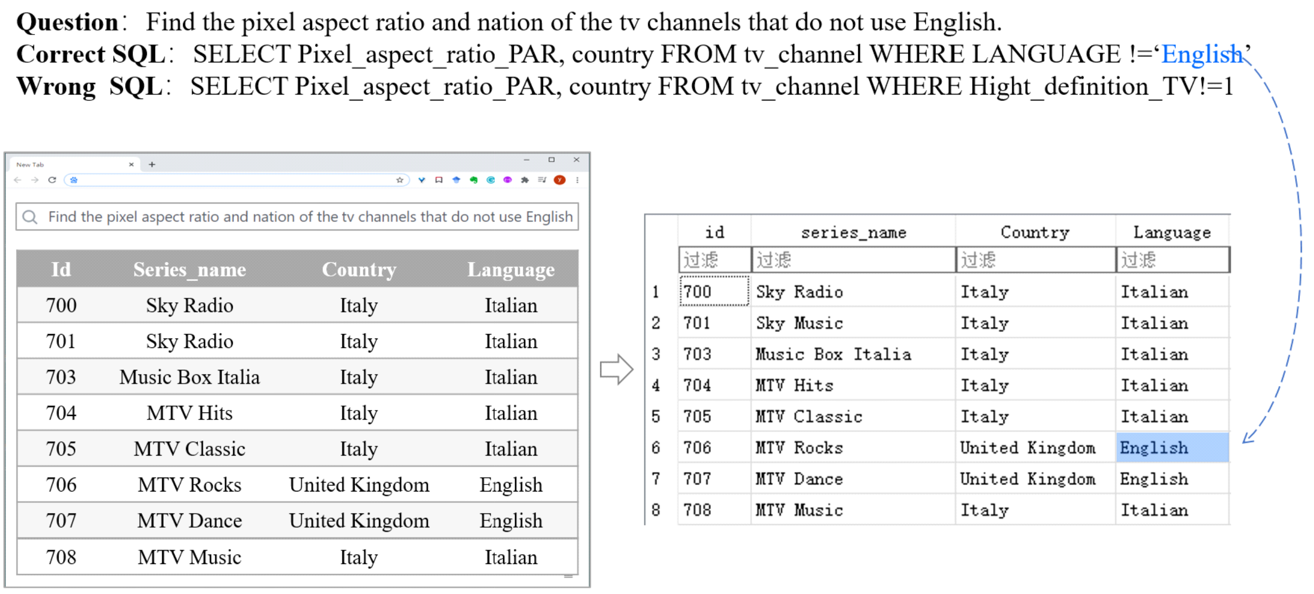
	Easy	Medium	Hard	Extra	All
IRNet	69.9%	55.5%	41.4%	30.4%	52.4%
Ours	70.8%	58.3%	44.7%	31.3%	54.6%

Table 3 shows the exact matching accuracy of IRNet and IRNet with our improvements. By adding database value to the model and using GGNN to encode the database structure, our methods improve the IRNet model by 2.2% overall. The table also shows the breakdown of prediction accuracy in groups of different difficulty level. Our approach clearly improves the accuracy of column and table prediction for easy, medium, hard and extra hard groups.

TABLE 4. An Ablation Study on Our Methods.

IRNet	with GGNN	with DB value	with GGNN+ DB value
52.4%	53.5%	53.3%	54.6%

To examine how each technique contributes to the performance, we conduct an ablation analysis (Table 4) of two aspects: 1) without GGNN, 2) without DB Value. Without GGNN, the model's exact matching accuracy decreases by



Question: Find the pixel aspect ratio and nation of the tv channels that do not use English.
Correct SQL: SELECT Pixel_aspect_ratio_PAR, country FROM tv_channel WHERE LANGUAGE !='English'
Wrong SQL: SELECT Pixel_aspect_ratio_PAR, country FROM tv_channel WHERE Hight_definition_TV!=1

Find the pixel aspect ratio and nation of the tv channels that do not use English

Id	Series_name	Country	Language
700	Sky Radio	Italy	Italian
701	Sky Radio	Italy	Italian
703	Music Box Italia	Italy	Italian
704	MTV Hits	Italy	Italian
705	MTV Classic	Italy	Italian
706	MTV Rocks	United Kingdom	English
707	MTV Dance	United Kingdom	English
708	MTV Music	Italy	Italian

	id	series_name	Country	Language
	过滤	过滤	过滤	过滤
1	700	Sky Radio	Italy	Italian
2	701	Sky Music	Italy	Italian
3	703	Music Box Italia	Italy	Italian
4	704	MTV Hits	Italy	Italian
5	705	MTV Classic	Italy	Italian
6	706	MTV Rocks	United Kingdom	English
7	707	MTV Dance	United Kingdom	English
8	708	MTV Music	Italy	Italian

FIGURE 7. An example of how our methods correct a column prediction error. The blue dashed line represents how the value in the database is mentioned in natural language query.

1.3%, and Without DB Value, the model's exact matching accuracy decreases by 1.1%. This means that using GGNN to encode DB Schema can effectively incorporate relationships between database structures into the model, and adding database value on top of the model improves the exact matching accuracy.

To study the performance of our methods in detail, we measure the average F1 score on different SQL components. We compare between IRNet and Ours. As shown in Fig. 6, Our methods outperform IRNet on SELECT, WHERE, GROUP BY and ORDER BY, but has a slightly decreased for AND/OR.

Fig. 7 presents an example to illustrate intuitively how our method rectify the error of column prediction with database value. As is shown, the column 'Language' is never referred in the natural language question, thus the original model generated the SQL with no column named 'Language' but a wrong column 'Hight_definition_TV'. With our approach of combining database value, the word "English" in the NL query is identified to be a potential database value. Then model then uses it to seek for the column it belongs to, and finally produces a correct SQL query statement. In contrast, an incorrect SQL query is generated without using our method.

D. DISCUSSION

To show the value of maximizing the use of information embedded in relational databases in order to improve the prediction performance of a TTS system, we have described following two new algorithmic components as extensions to the IRNet neural model:

1) Introducing database values into the model, computing the similarity between natural language or textual questions or queries and the database values, and establishing correlations between database values and column names through an Attention mechanism.
2) Using Gated Graph Neural Network (GGNN) to encode complete database schema, not only including tables and columns but also their relations;

The preliminary experimental results have demonstrated the improvement over the original IRNet implementation, and the potential value of our approach to enhance web-based information retrieval capabilities.

Some of the limitations of our approach are also clearly observed in our analysis and evaluation of the experimental results.

1) TABLE AND COLUMN PREDICTION

Our method cannot predict the correct column in approximately 50% of all analyzed errors. In about 20% of errors, it selects a column from another table, so the table's prediction is also wrong. The main reason for these errors is that the columns in different tables have similar names, so it is difficult to distinguish. Examples of such column names often appear in multiple tables. Incorporating more appropriate schema linking methods (for example, embedding-based methods rather than string-based methods used in IRNet) may help reduce such errors.

2) SQL SKETCH PREDICTION

In about 33% of the cases, we found errors in the SQL sketch. However, it is worth noting that the majority (69%) of these errors occurred in queries classified as Hard or Extra Hard in the Spider dataset. Some difficulties and special situations require advanced common sense, which is difficult to incorporate into the model. However, some examples of errors with lower difficulty may be easily solved with domain knowledge.

IV. CONCLUSION

In this article, we have discussed two improvements to the NL-SQL model in IRNet for web-based data and information retrieval. First, we introduced a representation of Gated Graph Neural Network to encode the database structure. Second, we include database values in our prediction model to compute the correlation between database values and column names, in order to alleviate the difficulties in matching column names due to lack of sufficient details in natural language queries. We train the revised model with the Spider dataset. The experimental results with the testing dataset empirically validate the merits of our model, and demonstrate its potential to gain performance improvement for web-based data and information retrieval. The method and algorithms discussed in this paper can be also applied to other types of DB based application systems. For future work, we will conduct further investigation of using natural language model(s) combined with application domain knowledge or semantic in developing TTS models and algorithms of higher performance.

REFERENCES

[1] J. Gemmell, G. Bell, and R. Lueder, "MyLifeBits: A personal database for everything," *Commun. ACM*, vol. 49, no. 1, pp. 88–95, Jan. 2006.

[2] S. Marcos-Pablos and F. J. García-Peñalvo, "Information retrieval methodology for aiding scientific database search," *Soft Comput.*, vol. 24, no. 8, pp. 5551–5560, Apr. 2020.

[3] M. L. Kersten, S. Idreos, S. Manegold, and E. Liarou, "The researcher's guide to the data deluge: Querying a scientific database in just a few seconds," *Proc. VLDB Endowment*, vol. 4, no. 12, pp. 1474–1477, Aug. 2011.

[4] *Web of Science*. Accessed: Dec. 25, 2020. [Online]. Available: http://apps.webofknowledge.com

[5] C. Tenopir and E. Read, "Patterns of database use in academic libraries," *College Res. Libraries*, vol. 61, no. 3, pp. 234–246, May 2000.

[6] V. S. Smith, "Data publication: Towards a database of everything," *BMC Res. Notes*, vol. 2, no. 1, pp. 1–3, Dec. 2009.

[7] T. Bai, Y. Ge, C. Yang, X. Liu, L. Gong, Y. Wang, and L. Huang, "BERST: An engine and tool for exploring biomedical entities and relationships," *Chin. J. Electron.*, vol. 28, no. 4, pp. 797–804, Jul. 2019.

[8] Z. Liu, J. Walker, and Y. Chen, "XSeek: A semantic XML search engine using keywords," in *Proc. 33rd VLDB*, Vienna, Austria, 2007, pp. 1330–1333.

[9] S. Oyama, T. Kokubo, and T. Ishida, "Domain-specific Web search with keyword spices," *IEEE Trans. Knowl. Data Eng.*, vol. 16, no. 1, pp. 17–27, Jan. 2004.

[10] J. X. Yu, L. Qin, and L. Chang, "Keyword search in databases," in *Synthesis Lectures on Data Management*, vol. 1, H. V. Jagadish, Ed. San Rafael, CA, USA: Morgan & Claypool Publishers, 2009, pp. 1–155. [Online]. Available: https://www.morganclaypool.com/doi/abs/10.2200/S00231ED1V01Y200912DTM001?journalCode=dtm

[11] V. Zhong, C. Xiong, and R. Socher, "Seq2SQL: Generating structured queries from natural language using reinforcement learning," 2017, *arXiv:1709.00103*. [Online]. Available: http://arxiv.org/abs/1709.00103

[12] X. Xu, C. Liu, and D. Song, "SQLNet: Generating structured queries from natural language without reinforcement learning," 2017, *arXiv:1711.04436*. [Online]. Available: http://arxiv.org/abs/1711.04436

[13] T. Yu, Z. Li, Z. Zhang, R. Zhang, and D. Radev, "TypeSQL: Knowledge-based type-aware neural text-to-SQL generation," in *Proc. Conf. North Amer. Chapter Assoc. Comput. Linguistics, Hum. Lang. Technol.*, vol. 2, 2018, pp. 588–594.

[14] L. Dong and M. Lapata, "Coarse-to-fine decoding for neural semantic parsing," in *Proc. 56th Annu. Meeting Assoc. Comput. Linguistics*, vol. 1, 2018, pp. 731–742.

[15] W. Hwang, J. Yim, S. Park, and M. Seo, "A comprehensive exploration on WikiSQL with table-aware word contextualization," 2019, *arXiv:1902.01069*. [Online]. Available: http://arxiv.org/abs/1902.01069

[16] J. Devlin, M. W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," in *Proc. NAACL-HLT*, Minneapolis, MN, USA, 2019, pp. 4171–4186.

[17] T. Yu, R. Zhang, K. Yang, M. Yasunaga, D. Wang, Z. Li, J. Ma, I. Li, Q. Yao, S. Roman, Z. Zhang, and D. Radev, "Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-SQL task," in *Proc. Conf. Empirical Methods Natural Lang. Process.*, Brussels, Belgium, 2018, pp. 3911–3921.

[18] T. Yu, M. Yasunaga, K. Yang, R. Zhang, D. Wang, Z. Li, and D. Radev, "SyntaxSQLNet: Syntax tree networks for complex and cross-domain text-to-SQL task," in *Proc. Conf. Empirical Methods Natural Lang. Process.*, Brussels, Belgium, 2018, pp. 1653–1663.

[19] D. Lee, "Clause-wise and recursive decoding for complex and cross-domain text-to-SQL generation," in *Proc. Conf. Empirical Methods Natural Lang. Process. 9th Int. Joint Conf. Natural Lang. Process. (EMNLP-IJCNLP)*, Hong Kong, 2019, pp. 6047–6053.

[20] B. Wang, R. Shin, X. Liu, O. Polozov, and M. Richardson, "RAT-SQL: Relation-aware schema encoding and linking for text-to-SQL parsers," in *Proc. 58th Annu. Meeting Assoc. Comput. Linguistics*, 2020, pp. 7567–7578.

[21] B. Bogin, J. Berant, and M. Gardner, "Representing schema structure with graph neural networks for text-to-SQL parsing," in *Proc. 57th Annu. Meeting Assoc. Comput. Linguistics*, Florence, Italy, 2019, pp. 4560–4565.

[22] Y. Li, D. Tarlow, M. Brockschmidt, and R. Zemel, "Gated graph sequence neural networks," 2015, *arXiv:1511.05493*. [Online]. Available: http://arxiv.org/abs/1511.05493

[23] J. Guo, Z. Zhan, Y. Gao, Y. Xiao, J.-G. Lou, T. Liu, and D. Zhang, "Towards complex text-to-SQL in cross-domain database with intermediate representation," in *Proc. 57th Annu. Meeting Assoc. Comput. Linguistics*, Florence, Italy, 2019, pp. 4524–4535.

[24] B. Carpenter, *Type-Logical Semantics*. Cambridge, MA, USA: MIT Press, 1998.

[25] R. Kate, Y. Wong, and R. J. Mooney, "Learning to transform natural to formal languages," in *Proc. AAAI*, Pittsburgh, PA, USA, 2005, pp. 1062–1068.

[26] P. Liang, M. I. Jordan, and D. Klein, "Learning dependency-based compositional semantics," *Comput. Linguistics*, vol. 39, no. 2, pp. 389–446, Jun. 2013.

[27] J. Berant, A. Chou, R. Frostig, and P. Liang, "Semantic parsing on freebase from question-answer pairs," in *Proc. EMNLP*, Seattle, WA, USA, 2013, pp. 1533–1544.

[28] P. Pasupat and P. Liang, "Compositional semantic parsing on semi-structured tables," in *Proc. 53rd Annu. Meeting Assoc. Comput. Linguistics 7th Int. Joint Conf. Natural Lang. Process.*, Beijing, China, vol. 1, 2015, pp. 1470–1480.

[29] T. Luong, H. Pham, and C. D. Manning, "Effective approaches to attention-based neural machine translation," in *Proc. Conf. Empirical Methods Natural Lang. Process.*, Lisbon, Portugal, 2015, pp. 1412–1421.

[30] A. Arasu and H. Garcia-Molina, "Extracting structured data from Web pages," in *Proc. ACM SIGMOD*, San Diego, CA, USA, 2003, pp. 337–348.

[31] D. W. Embley, C. Tao, and S. W. Liddle, "Automating the extraction of data from HTML tables with unknown structure," *Data Knowl. Eng.*, vol. 54, no. 1, pp. 3–28, Jul. 2005.

[32] W. Gatterbauer, P. Bohunsky, M. Herzog, B. Krüpl, and B. Pollak, "Towards domain-independent information extraction from Web tables," in *Proc. 16th Int. Conf. World Wide Web (WWW)*, Banff, AB, Canada, 2007, pp. 71–80.

[33] L. R. Lautert, M. M. Scheidt, and C. F. Dorneles, "Web table taxonomy and formalization," *ACM SIGMOD Rec.*, vol. 42, no. 3, pp. 28–33, Oct. 2013.

[34] S. Zhang and K. Balog, "Web table extraction, retrieval and augmentation," in *Proc. 42nd Int. ACM SIGIR Conf. Res. Develop. Inf. Retr.*, Paris, France, Jul. 2019, pp. 1409–1410.

[35] M. Najork, "Web crawler architecture," in *Encyclopedia of Database Systems*, L. Liu and M. T. Özsu, Eds. Berlin, Germany: Springer, 2009, pp. 1–3. [Online]. Available: https://www.morganclaypool.com/doi/abs/10.2200/S00231ED1V01Y200912DTM001?journalCode=dtm

[36] M. Thelwall, "A Web crawler design for data mining," *J. Inf. Sci.*, vol. 27, no. 5, pp. 319–325, Oct. 2001.

[37] W. McKinney, "Pandas: A foundational Python library for data analysis and statistics," *Python High Perform. Sci. Comput.*, vol. 14, no. 9, pp. 1–9, Nov. 2011.

[38] W. Cavnar and J. Trenkle, "N-gram-based text categorization," in *Proc. SDAIR 3rd Annu. Symp. Document Anal. Inf. Retri.*, 1994, pp. 161–169.

[39] K. Cho, B. van Merrienboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using RNN encoder–decoder for statistical machine translation," in *Proc. Conf. Empirical Methods Natural Lang. Process. (EMNLP)*, Doha, Qatar, 2014, pp. 1724–1734.

[40] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, "Pytorch: An imperative style, high-performance deep learning library," in *Proc. Adv. Neural Inf. Process. Syst.*, 2019, pp. 8026–8037.

[41] J. Pennington, R. Socher, and C. Manning, "Glove: Global vectors for word representation," in *Proc. Conf. Empirical Methods Natural Lang. Process. (EMNLP)*, Doha, Qatar, 2014, pp. 1532–1543.

[42] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014, *arXiv:1412.6980*. [Online]. Available: http://arxiv.org/abs/1412.6980

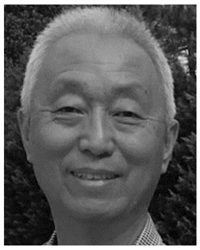
[43] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *J. Mach. Learn. Res.*, vol. 15, no. 1, pp. 1929–1958, 2014.

TIAN BAI (Member, IEEE) received the Ph.D. degree from Jilin University, Changchun, China, in 2012. He is currently an Associate Professor with the College of Computer Science and Technology, Jilin University. His research interests include bioinformatics and machine learning.

YAN GE is currently pursuing the master's degree with the School of College of Software, Jilin University, Changchun, China. His research interests include machine learning and NLP.

SHUYU GUO is currently pursuing the Ph.D. degree with the College of Computer Science and Technology, Jilin University, Changchun, China. His research interests include machine learning and NLP.

ZHENTING ZHANG is currently pursuing the master's degree with the College of Computer Science and Technology, Jilin University, Changchun, China. Her research interests include machine learning and NLP.



LEIGUANG GONG received the Ph.D. degree from Rutgers University, NJ, USA, in 1992. He was a Senior Researcher with the IBM Watson Research Center before retirement. He is currently an Advisor of Yantai Huashen Intelligent Technology Limited, Yantai, China.