RGANN: Advancing CPC Patent Classification with Novel Multi-Label Datasets

Yuan Meng, Shen Nie, Qiao Jia, Ye Yuan*, Xuhao Pan

School of Statistics and Data Science
Shanghai University of International Business and Economics
Shanghai, China

nancymeng@suibe.edu.cn, 3055155944@qq.com, 357469227@qq.com, yuany@suibe.edu.cn, kankanxiaopan@163.com

***Abstract*—As the number of patent applications is increasing sharply, the CPC system is gradually replacing the IPC system as the new standard. However, due to the lack of public datasets, there are still some research gaps in the study of automatic patent classification based on the CPC system. In this work, we first propose two new patent datasets on multi-label classification using CPC, named Patent14K and Patent2M. Then, we introduce a series of baseline models based on BERT-for-Patents with the latest optimization methods on our dataset. Finally, we propose an innovative model, named Relation Graph Attention Neural Network (RGANN) based on the syntactic graph structure to improve the baseline models. Experimental results demonstrate that RGANN achieves a large F1-score improvement (3.01% on Patent14K dataset, 4.22% on Patent2M dataset) compared to the baseline models. Our all codes, datasets and the way to reproduce our results are available at https://github.com/qingtian5/RGANN4Patent2M.**

Keywords—RGANN, Syntactic Graph, Patent14K, Patent2M, CPC classification.

I. INTRODUCTION

Patent classification is a multi-label classification task, which is very challenging because the number of labels may be large, even exceeding 600 subclass levels. In this paper, we promote this task from the following three perspectives: datasets, deep learning baseline models with optimization methods, and model improvement.

From the perspective of patent datasets, it is time to build a large-scale dataset based on the CPC (Cooperative Patent Classification) system for future related research on patent classification. It is well known that the CPC system and the IPC (International Patent Classification) system are two of the most commonly patent classification systems. In comparison, CPC is a more specific and detailed version than the IPC. On 1st January 2013, the CPC system went into effect and the United States Patent and Trademark Office (USPTO) replaced the previous classification system with it. Since then, more and more national patent offices decided to follow the CPC [1]. However, public patent datasets based on CPC for research are extremely scarce. Due to the 2011 CLEF-IP competition [2], most of the papers in this field are based on patents filed between 1978 and 2009 with IPC system at the subclass level. In general, key performances are evaluated in P@1 (top 1 for precision), P@5, R@5 (top 5 for recall), F1@5 and other metrics. But it is very confusing that R@1 and F1@1 were omitted in the above-mentioned papers. Metric selection is critical to our multi-label classification work, since precision values can be very high at the expense of very low recall. So, P@1 alone may not be a fair number if R@1 and F@1 are not provided. In our work, we prepare a new large-scale dataset based on CPC with over 2 million patents and a new small dataset based on CPC with 14,000 training and 4000 testing patents. Patent researchers can leverage these datasets or our approach to cover more tasks. Meanwhile, considering that F1@1 is the best evaluation metric for patent classification, we use F1@1 as our main evaluation metric. P@1 and R@1 are also used as auxiliary evaluation metrics. Because the latest AI technology is developing very fast, the appearance of many new patents and new CPC classification numbers will make the distribution of the latest patent data very different from the previous ones, which is a huge and very real challenge. Therefore, we reasonably arrange the latest data as a test set. In addition, it is important to note that our datasets are based on patent claims. The reason why the claims in the patent are selected in our datasets is based on the following three considerations. First, the importance of patent claims has been underestimated in the past. For every patent, drafting the patent claims is the first step. The purpose of patent claims is to set the scope of an invention. The rest of the patent document can be derived or expanded from the claims, which patent practitioners typically develop when drafting a new patent application. Second, within the realm of patent law, the scope of a patented invention is established by its claims, which delineate its boundaries. A "fundamental tenet" of patent law is that the patentee is entitled to exclude others from utilizing the inventions that are defined by the claims of the patents [3]. Third, the primary rationale for employing patent claims is to facilitate subsequent downstream tasks related to generating such claims. Our study, to the best of our knowledge, represents the inaugural effort to concentrate exclusively on patent claims, specifically the abstract textual content, as opposed to employing claims as ancillary data in prior work.

From the perspective of deep learning baseline model, pre-training an unsupervised language model on large corpus and fine-tuning model on downstream tasks have recently achieved several state-of-the-art performances. Such pre-trained models include ULMFiT [4], BERT [5], XLNet [6] and OpenAI GPT-3 [7]. Among them, BERT is the most suitable for experiments if the availability of source code and the availability of pre-trained models are considered. In order to balance and compare the BERT model, we also select several relatively important models that recently optimize the output from BERT – like Bert-flow [8], Bert-whitening [9], and SimCSE [10] for experiments.

From the perspective of model improvements, we also propose a new graph embedding model that uses syntactic structure information——Relation Graph Attention Neural

*Corresponding author: yuany@suibe.edu.cn

Network (RGANN) to update the embeddings representation of BERT output. The model we proposed achieves better performance compared with original BERT model.

Our main contributions are shown below:

(a) We provide two CPC-based classification benchmark datasets for patent multi-label classification.

(b) We provide a baseline based on BERT and the latest optimization methods on the two datasets.

(c) We propose a RGANN model to optimize the embedding representation of BERT using syntactic structure, and obtain comparable results.

II. Related Work

In recent years, due to the rapid development of deep learning, the research on patent classification has also made significant progress. Grawe [11] proposes a model combining word2vec with LSTM in patent IPC classification to obtain 63% accuracy on the CLEE-IP dataset. Li [12] applies DeepPatent to the patent IPC classification. It achieves 83.98% accuracy on the CLEE-IP dataset and 73.88% accuracy on the USPTO-2M dataset. Yu [13] proposes a reinforcement learning-based patent IPC classification structure for TRIZ. Shalaby [14] uses LSTM to enhance the document representation, which also greatly improves the accuracy of patent IPC classification. Lee [15] proposes BERT-for-Patents as a large pre-training model on patent data. We find that most of the patent classification tasks done by previous authors were based on IPC, and ignored the importance of CPC, so we propose two patent datasets on the CPC multi-label classification task. Another observation is that there are also some models optimized for the sentence embedding representation of BERT in recent years, like BERT-flow, SimCSE, etc. Therefore, we also conduct experiments on these models. Further, we propose a new module (RGANN) to optimize the BERT embedding representation to learn the syntactic structural embedding of the text. Our RGANN model achieves comparable results on our proposed datasets.

A recent observation is the development of large language models (LLM). Recently, the power of large language models is enormous, like ChatGPT (base GPT-4) [16], and they can do well for different downstream tasks. However, Jiao [17] and Hendy [18] compare the translation capabilities of ChatGPT with the current best models for translation tasks, and find that ChatGPT does not perform better at translation tasks than other models. We try different prompt ways to help ChatGPT (base GPT-4) to do patent multi-label classification task. As a result, we find that ChatGPT (base GPT-4) performs poorly on fine-grained task of patent multi-label classification. Hence, we leave how to do in-context learning to make large language models (LLM) better performance at specific tasks to the future research.

III. Datasets

Most patent datasets in the past have come from CELF-IP or the patent office. In contrast, we utilize the Google Patent Public Dataset [19] on BigQuery, released in 2017. The development of AI technology has caused the potential distribution of patents to change dramatically since 2012, so we collect all the latest relevant patents from 2012-2018 (2059636 in total) in BigQuery using SQL code (The SQL code we used is available in our code link). This data is used as training dataset, and 20,000 patents are carefully selected from 2019 in BigQuery as test dataset. We combine the training data and testing data as the dataset, named Patent2M. We explain more details about sample data in TABLE I.

TABLE I. Sample Data.

Column Name	Example	DataType
cpc_ids	B64C, Y02T	str, CPC category
id	8087618	str
date	2012-01-03	str
text	1. A propulsion system…	str, patent claim

We also study the distribution of the data in detail. And in Fig. 1, we can clearly see that our training data shows a long-tailed distribution. A very small number of CPC categories each has more than 50k patent texts matched with it. We consider them to be general patent categories. There is also a large portion of the 600+ patent categories that have very few matches, and we consider this CPC category to be an intradomain patent category. We show our CPC classification numbers and their corresponding number of patent texts about Patent2M dataset in detail in TABLE II. For this case, the difficulty of our patent classification model is how to correctly identify the tail patents.

TABLE II. Sample CPC and its number in Patent2M.

CPC(Max num)	Number	CPC(Min num)	Number
G06F	287338	…	…
H04L	200973	G21Y	2
H01L	170760	G21J	2
Y10T	149665	C06F	1
…	…	F21H	1

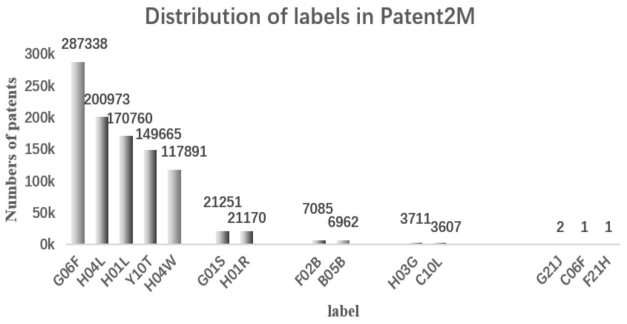

Fig. 1. Distribution of different label after sorting by number in Patent2M.

In the real case, we may not have that much data to train the model. The real data we can collect may be very small, so we also provide a small dataset for classification with CPC, named Patent14K dataset. This dataset is collected from five thousand patent data for each of the years 2012 to 2018 retrieved in BigQuery, which will be used as training dataset. Five thousand pieces of patent data from 2019 in BigQuery are collected as test dataset. At the same time, in order to keep the same distribution with the original Patent2M dataset, we perform a detailed manual inspection of the five thousand patents for years 2012 to 2019. Finally, two thousand pieces of data for each of the years 2012 to 2018 are screened out as training dataset and four thousand pieces of data for 2019 as test dataset. In total, there are 14,000 training entries, so the dataset is named the Patent14K dataset. As shown in Fig. 2, our Patent14K dataset also has a long-tailed distribution. We also show our CPC classification numbers and their

corresponding number of patent texts about Patent14K dataset in detail in TABLE III.

TABLE III. SAMPLE CPC AND ITS NUMBER IN PATENT14K.

CPC(Max num)	Number	CPC(Min num)	Number
G06F	287338	…	…
H04L	200973	G21Y	2
H01L	170760	G21J	2
Y10T	149665	C06F	1
…	…	F21H	1

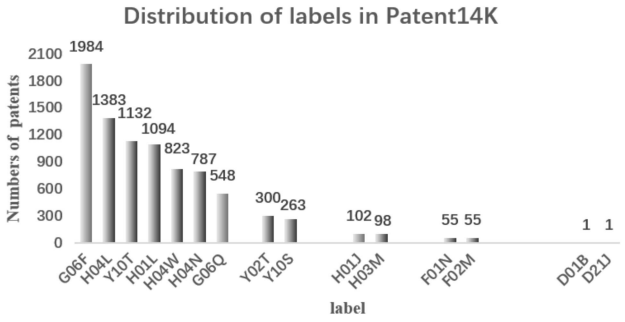

Fig. 2. Distribution of different label after sorting by number in Patent14K.

IV. METHOD

We have mentioned the BERT model in the introduction, and previous work is based on BERT. However, we believe that there is a mismatch between the text used for the pre-training process of BERT and the patent text. There are a lot of patent terms in the patent text. In fact, BERT does not capture the distribution of patents well in the pre-training process, so it does not achieve a good result in the patent classification task.

Google released a large BERT model pre-trained on patent data in 2020 [20]. The BERT model has been trained on >100 million patent documents and was trained on all parts of a patent (abstract, claims, description). It has a similar configuration to the BERT-Large model, such as the maximum input sequence length is 512 tokens and maximum masked words for a sequence is 45. The vocabulary has approximately 8000 added words from the standard BERT vocabulary. These represent frequently used patent terms. The vocabulary includes "context" tokens indicating what part of a patent the text is from (abstract, claims, summary, invention). This model will be named as BERT-for-Patents model in the following. The BERT-for-Patents model is available in the link (https://huggingface.co/anferico/bert-for-patents), so we use this model easily by importing the transformers package.

We also experimented with some of the latest optimization methods for the BERT model. In addition, we propose a new optimization method for the BERT model, which results in a significant improvement over the original BERT-for-Patents on both the Patent2M and Patent14K datasets.

A. BERT-for-Patents baseline

We utilize the published BERT-for-Patents pre-training model (340M parameters). For multi-label classification, we use a sigmoid cross-entropy with logits function to replace the original softmax function, which is suitable for one-hot classification only. We intentionally keep the code changes as small as possible in order to make the BERT-for-Patents test a basic benchmark for our and future experiments.

As shown in Fig. 3, we use the average pooling of final outputs without [cls] token, and add a MLP layer (including two fully connected layers and BatchNorm) for the task of multi-label classification. And This is a common way to use BERT for downstream tasks.

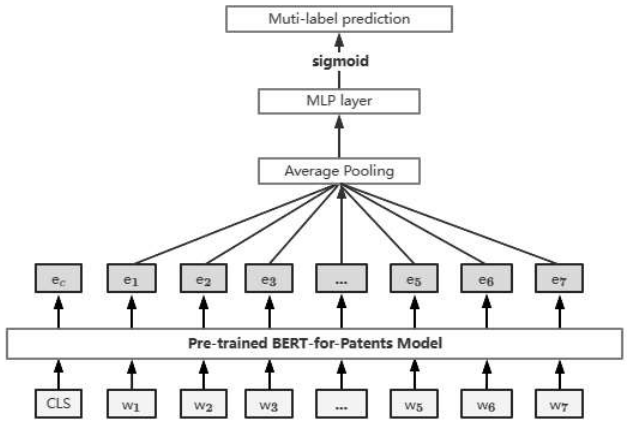

Fig. 3. A baseline approach to patent multi-label classification using BERT-for-Patents.

After that, we will not discuss the [cls] token because our ablation studies in Section V show that using only [cls] token for the downstream task is much less effective than using average pooling. And we will refer to this embedding after average pooling as **AVG Embedding** in the following. The **AVG Embedding** is representing the meaning of the whole sentence.

B. Some optimization methods for BERT-for-Patents outputs

BERT-flow [8]. Due to the limited space of this paper, we only briefly describe the general flow of the existing optimization algorithm for Bert. Since the AVG Embedding representing the whole sentence learned by BERT is not uniformly distributed over the vector space, BERT-flow learns an invertible mapping to map BERT AVG Embedding space to a smooth, isotropic standard Gaussian distribution. A simple way for us to apply BERT-flow to BERT-for-Patents model is to add an invertible transformation before the MLP layer, as follows:

$$p_u(\mathbf{u}) = p_Z\left(f_\phi^{-1}(\mathbf{u})\right)\left|\det\frac{\partial f_\phi^{-1}(\mathbf{u})}{\partial \mathbf{u}}\right| \quad (1)$$

where p_Z is the standard Gaussian distribution, $\mathbf{u}$ is the distribution of AVG Embedding. We need to maximize the pre-computed likelihood function of the BERT-for-Patents [AVG Embedding] token vector:

$$D\sim\left(\log p_Z\left(f_\phi^{-1}(\mathbf{u})\right)+\log\left|\det\frac{\partial f_\phi^{-1}(\mathbf{u})}{\partial \mathbf{u}}\right|\right) \quad (2)$$

$$\max_\phi \mathbb{E}(\mathbf{u}=\text{BERT-for-Patents}([\text{AVG}]\sim\mathcal{D})) \quad (3)$$

where f_ϕ is a reversible neural network. The above formula simplifies the design of the Glow model [21] by replacing the affine coupling transform of Glow with an additive coupling transform and the **1x1** convolution with a random permutation. Finally, we learn a bidirectional mapping

transformation f_{ϕ}^{-1}. It transforms the BERT-for-Patents AVG Embedding vector **u** into a potential Gaussian representation **z** with no loss. The transmission of semantics with no loss is guaranteed by the invertibility of f_{ϕ}.

BERT-whitening [9]. Compared with BERT-flow, BERT-whitening does a PCA [22] operation on the AVG Embedding, reducing the dimension of the AVG Embedding of BERT. For BERT-for-Patents model, we also only need to add a whitening (as same as PCA) operation before the MLP layer.

SimCSE [10]. This work uses a self-supervised approach to improve the AVG Embedding representation of the model. Different droupout rates are used to construct the positive examples. Passing a sample through the encoder twice, a positive example pair is selected and negative examples are other sentences in the same batch. For the BERT-for-Patents model, we simply change the dropout rates parameter for a few epochs of self-supervised training.

C. RGANN methods for BERT-for-Patents outputs

There is a lot of evidence that syntactic structure improves the performance of the natural language model significantly in a range of downstream NLP tasks. Xu [23] used RGCN to help BERT model better perform the task of reference resolution and classification.

Since BERT-for-Patents model cannot learn the syntactic structure information of the text well, and the syntactic structure information is also very important for downstream tasks. Therefore, we design a relational graph attention neural network (RGANN) that digests syntactic structure information to further optimize the embedding representation of BERT-for-Patents.

We extract the syntactic structure information of the patent text by using the SpaCy module. We use the extracted information to construct the syntactic graph structure data, as shown in Fig.4. For every black square, it is a token.

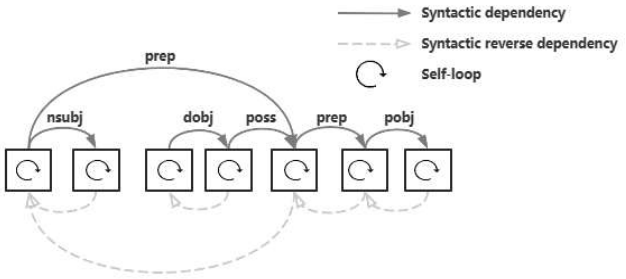

Fig. 4. Information Flows in Syntactic Dependency Graph

The core idea of GATNE-T (we refer to this as GATNE in the paper) is to aggregate neighbors of different edge types to the current node and then generate different vector representations for nodes of each edge type. Inspired by the GATNE algorithm proposed by Cen [24], we adjust GATNE and propose the Relational Graph Attention Neural Network (RGANN) applied in the syntactic dependency graph with multiple edges to learn uniform embeddings.

First, GATNE obtains the embedded representations of nodes and edges on a large graph structure data, but our RGANN model needs to adapt to different syntactic graph structures generated by different samples. Therefore, a new attention architecture is proposed to solve this problem. Second, the initial embedding representations of GATNE are randomly generated, but our goal is to solve the ambiguous pronouns coreference resolution, thus it is natural to use BERT Embeddings to initialize the node embeddings. Third, in order to take advantage of all the information, we add a shortcut module in the model so that our initialization node embeddings can also be concatenated into the final output embeddings.

Specially, the RGANN model splits the overall embedding of a certain node v on each edge type i in the syntactic dependency graph into two parts: base embedding and three edge embeddings. The base embedding of node v is shared between its three different edge types. We use BERT-for-Patents Embedding as the base embedding of the nodes. We follow the following aggregation steps to obtain the final syntactic embeddings of each node.

First, the representation of each node is compressed to obtain a more compact representation denoted as u_i^{base}, which is used as the base embedding.

$$u_i^{base} = W_{r0} u_i^{out} \tag{4}$$

where $W_{r1} \in R^{1024*256}$ is learnable, and u_i^{out} is the BERT-for-Patents representation of node v on edge type i. In our work, the representation of each node v from BERT-for-Patents is a vector of 1024 dimensions. Consistent with previous work [23], the compressed node representation dimensions are set to 256. So u_i^{base} is a vector of 256 dimensions.

Second, following GraphSage [25], we obtain each type of edge embedding for node v by aggregating from neighbors' edge embeddings. We randomly sample n neighbor nodes for each edge embedding $u_{j,r}^{base}$, and then aggregate them. The aggregator function can be a sum, mean or max-pooling aggregator as follows:

$$U_{i,r} = W_{r1}\,\text{aggregator}\left(\left\{u_{j,r}^{base}, \forall u_j \in N_{i,r}, j = 0, \dots, n\right\}\right) \tag{5}$$

where $W_{r1} \in R^{d*m}$, is a learnable parameter, d is 256, m is the hyperparameter that needs to be given. In order to make the attention calculation more convenient, we compress the aggregated representation again.

Third, applying the attention mechanism to get the weight of each aggregated edge representation for each node as follows:

$$a_{i,r} = \text{softmax}\left(w_r^T \tanh\left(W_{r2} U_{i,r}\right)\right)^T \tag{6}$$

where $w_r \in R^n, W_{r2} \in R^{n*m}$, are learnable parameters, n is the hyperparameter that needs to be given.

Fourth, combining each weighted aggregated representation with the base embedding, the final representation of each node in edge type r can be represented as $v_{i,r}$, which is denoted as follows:

$$v_{i,r} = u_i^{base} + a_{i,r} M_r U_{i,r} \tag{7}$$

where $M_r \in R^{d*m}$ is a learnable parameter, u_i^{base} is base embedding and $v_{i,r}$ is a vector with 256 dimensions.

Finally, the syntactic embedding representation of each node is aggregated from three kinds of node representation $v_{i,0}, v_{i,1}, v_{i,2}$, denoted as follows:

$$v_i = \text{aggregator}(v_{i,0}, v_{i,1}, v_{i,2}) \quad (8)$$

where the aggregator function is a concatenate operation. It is important that, before the average pooling, we use a graph neural network to optimize each token. For a further fair comparison, we report our results in ablation studies.

D. Overall Model Architecture

The previous RGCN model [23] only uses the information of neighbor nodes, but it brings a significant improvement in coreference resolution tasks. Therefore, we think the potential of learning syntactic structure using RGANN can be much more than that. The most important reason is that by designing attention mechanisms, we obtain more valuable information from different types of edges.

Fig. 5 explains in detail how to extract Syntactic Embeddings from information flows in syntactic dependency graph and BERT Embeddings.

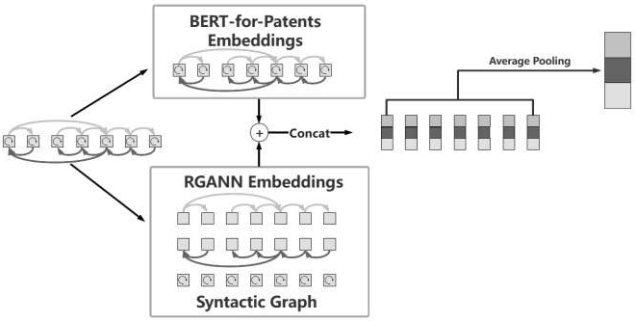

Fig. 5. Embedding Structure of Syntactic Dependency Graph.

For one thing, we use BERT-for-Patents Embeddings as the base embeddings. For another thing, we use three different types of syntactic relations to construct the syntactic relation graph with attention information to learn RGANN embeddings. In order to retain information from BERT-for-Patents Embeddings, we concatenate the embeddings that represent the relation graph with attention information and BERT-for-Patents Embeddings. Then, we concatenate different embeddings from three different kinds of edges (different color represent embeddings from different edge types). Finally, the average pooling of each word embedding is named as Syntactic Embedding.

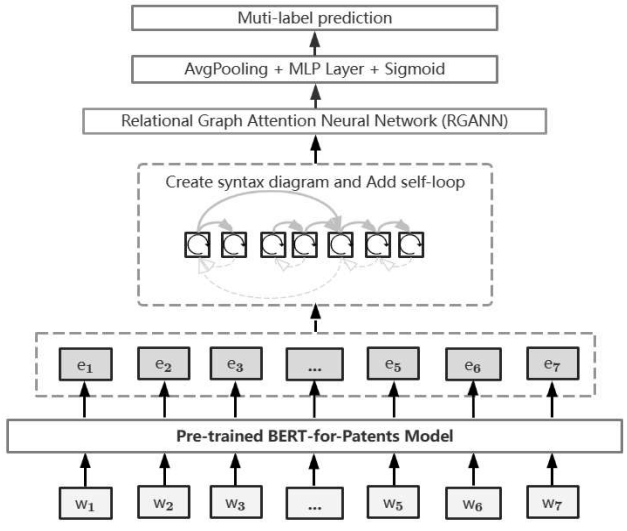

Fig. 6. Our proposed overall architecture of RGANN+BERT-for-Patents for patent multi-label classification.

We blend the syntactic embedding derived from the syntactic dependency graph with the pretrained BERT-for-Patents embeddings by connecting BERT-for-Patents embedding and syntactic embedding in series. This integrated architecture can help us learn better-performing embeddings when dealing with the task of pronoun resolution.

We first use the pre-trained BERT-for-Patents to extract context information between words and then connect with syntactic information from RGANN to form a "look again" mechanism to further obtain blending representations that are more beneficial to the current task. The specific architecture is shown in Fig. 6.

As shown in Fig. 6, the pre-trained BERT-for-Patents obtains the hidden feature representation, then RGANN looks at the syntactic information of the sentence again. Relying on the syntactic information derived from RGANN, we can obtain the hidden state in the sentence. It is used to get a more compact embedding representation for each word.

E. Pseudo code

We also provide our algorithm pseudo code, as shown in the TABLE IV. It succinctly outlines our training process and provides a detailed description of each step in our inference method.

TABLE IV. PSEUDO CODE

Algorithm 1: RGANN after BERT-for-Patents

Input: A long string of text (text) and the number of categories (N).

Output: A tensor only contains 0 and 1, and the length of the tensor is N.

```
bert = BERT("bert-for-patents")
rgann = RGANN()
tokenizer = Tokenizer()
# traning
bert.train(), rgann.train()
train(bert, rgann, N, datasets)

# get output from input
bert.eval(), rgann.eval()
# get the output result of the BERT-for-Patents
token = tokenizer(text)
bert_output = bert(token).hidden_states[-1]
# get the output result of the RGANN
rgann_output = rgann(bert_output)
output = MLP(torch.mean(rgann_output, dim=1))
# get the final prediction result, the shape is (B, N)
pred = torch.sigmod(output)
# get the categories from the pred tensor
pred_ categories = decode(pred)
```

V. EXPERIMENT

A. Datasets

TABLE V. ABLATION STUDIES

	Method	Dataset	F1 (%)	P (%)	R (%)
(a)	BERT-for-Patents+[cls]	Patent14K	59.18	60.86	65.73
(b) (baseline)	BERT-for-Patents+avgpooling	Patent14K	**60.52**	**61.76**	**66.23**
(c)	BERT-for-Patents+GCN	Patent14K	61.83	63.53	67.11
(d)	BERT-for-Patents+RGANN w/o R	Patent14K	63.02	63.76	68.82
(e)	BERT-for-Patents+RGANN w/ R	Patent14K	**63.53**	**64.15**	**69.46**

TABLE VI. MUTI-LABEL CLASSIFICATION PERFORMANCE

	Method	Dataset	F1 (%)	P (%)	R (%)
(a)	BERT-for-Patents+BERT-flow	Patent14K	59.23	60.45	65.43
(b)	BERT-for-Patents+BERT-whitening	Patent14K	59.46	61.47	66.12
(c)	BERT-for-Patents+SimCSE	Patent14K	60.78	62.10	66.56
(d) (baseline)	BERT-for-Patents+avg pooling	Patent14K	60.52	61.76	66.23
(e)	BERT-for-Patents+RGANN (ours)	Patent14K	**63.53**	**64.15**	**69.46**
(f) (baseline)	BERT-for-Patents+avg pooling	**Patent2M**	62.12	64.33	70.74
(g)	BERT-for-Patents+RGANN w/ R	**Patent2M**	**66.34**	**68.86**	**73.96**

Experiments are conducted using our own collected and collated Patent2M and Patent14K datasets, and detailed experimental results are reported for Patent14K. Because of computing power limitations, we only use the BERT-for-Patents baseline model and our proposed RGANN+BERT-for-Patents model on the Patent2M for experiments. Further exploration is left to future research.

B. Implementation Details

To enhance the model's capacity for generalization, we implemented a 5-fold cross-validation strategy to partition the training dataset into five equally sized subsets. In this iterative process, each experiment involves utilizing a distinct subset for validation while the remaining subsets are harnessed for training purposes. The model parameters that yield the most promising performance on the validation set are consequently utilized on the test set to generate the predicted outcomes.

In this manner, a total of five prediction outcomes are achieved, each one corresponding to a unique training-validation split. As an aggregate measure of these individual results, the mean value is computed and employed as the final prediction result. This methodology not only mitigates the risk of overfitting but also ensures a more robust evaluation of the model's performance, since it effectively utilizes the entirety of the dataset for both training and validation, while also providing multiple assessments of the model's predictive capability.

Furthermore, the iterative nature of this strategy introduces an element of redundancy, which bolsters the robustness of the model's evaluation by ensuring that its performance is not unduly influenced by any single split of the data. This leads to a more comprehensive and reliable understanding of the model's capacity to generalize its learned patterns to unseen data, thus enhancing the validity and reliability of our findings.

C. Evaluation Metrics

To utilize the pre-trained model, we do not change the hyperparameters of BERT-for-Patents model. We also freeze the whole model parameters of the BERT-for-Patents model, and we only use it as a feature extractor. We use a five-fold cross-validation on the training dataset and train all parts for 30 epochs, that means for one model we need to train 150 epochs. We use the AdamW [26] optimizer with a learning rate of 5e-4. We treat the output after sigmoid layer greater than k as the labels predicted by the model. Here we take k to be 0.25. All experiments were done on NVIDIA RTX A5000 GPU, and we use PyTorch1.11.0, Python3.8, and Cuda11.3 as our experimental environment.

For our multi-label classification task, we provide the best and the most commonly metrics – Precision(P@1), Recall(R@1), and F1-score(F1@1). We will briefly describe how Precision, Recall, and F1-score values are calculated in a multi-label classification task.

Precision is calculated as the average precision rate of all samples. For each sample, the precision is the number of correctly predicted labels as a percentage of the number of labels predicted as correct for the whole classifier. As follows, y_{true} represents the true value, y_{pred} represents the predicted value:

$$Precision = \frac{1}{N}\sum \frac{|y_{true} \cap y_{pred}|}{|y_{pred}|} \tag{9}$$

Recall is calculated as the average precision rate over all samples. And for each sample, the recall is the number of correctly predicted labels as a percentage of the overall number of true labels. As follows:

$$Recall = \frac{1}{N}\sum \frac{|y_{true} \cap y_{pred}|}{|y_{true}|} \tag{10}$$

F1-score is the summed mean of precision and recall, as follows:

$$F_1 - score = \frac{1}{N}\sum \frac{2Precision * Recall}{Precision + Recall} \tag{11}$$

D. Ablation Studies

We conduct detailed ablation experiments to verify the effectiveness of the syntactic graph structure for multi-label classification task, as shown in TABLE V. In row (a) and (b), we compare the results of using AVG Embedding and [cls] token. We use the results of AVG Embedding as our baseline. In row (c), because we construct the syntactic graph structure, we use the most classical graph neural network -- GCN to optimize each token for classification task. Compared with

row (a), GCN can take a little F1-score improvement (1.31%). In row (d), RGANN w/o R represents that in our model we do not consider the type of edges. In row (e), we consider three different type of edges and obtain greater improvement (3.01% by baseline). On the patent multi-label classification task, which has little relation with syntactic structure, our results also show the great effectiveness of syntactic structure.

E. Performance Evaluation

To evaluate the performance of the proposed modules, BERT-for-Patents [15], BERT-flow [8] + BERT-for-Patents, BERT-whitening [9] + BERT-for-Patents, SimCSE [10] + BERT-for-Patents, RGANN + BERT-for-Patents are used for our muti-label classification task. The comparison of results is shown in TABLE VI.

In TABLE VI, we show the original fine-tuning performance of BERT-for-Patents in row (d). In row (a) (b) (c), on the patent multi-label classification, we show the performance changes based on some of the latest modules optimized for BERT. In row (e), we show the improvement of our proposed RGANN module using syntactic graph structure for patent multi-label classification. In row (f) (g), we show the benchmark results on a large Patent2M dataset and the improvement results of our proposed RGANN module.

We compare the results of row (a) (b) (c) with row (d) and find that BERT-flow and BERT-whitening have some decreases for the results. Although BERT-flow and BERT-whitening optimize the embedding representation, they lose some information that is very useful for downstream tasks, so they are not very effective. In contrast, SimCSE does not reduce the dimensionality of the token and it optimizes token distribution, so it is helpful for downstream tasks. Comparing the results of row (d) and row (e), we find that our proposed module RGANN for learning syntactic graph structure can bring a relatively large F1-score improvement (3.01%). Also looking at the results of row (f) and (g), we find that our RGANN module trained on a larger Patent2M dataset obtains a larger F1-score improvement (4.22%). This shows the great potential of our RGANN module. If a larger model or a larger dataset is available, we believe that our proposed RGANN module can bring a comparable result.

VI. Conclusion and Discussion

A. Conclusion

In the pursuit of advancing patent categorization using Cooperative Patent Classification (CPC), we have introduced a substantial benchmark dataset, named Patent2M. This dataset is meticulously designed for multi-label classification tasks, aiming to provide a new standard against which future methods can be evaluated. Recognizing the necessity for testing under constrained conditions, we have also proposed a smaller, more focused dataset - Patent14K, specifically contrived for the same CPC multi-label classification challenge. These two distinct datasets, we believe, offer a comprehensive testbed that accurately reflects a broad array of realistic scenarios in patent classification.

Keeping in pace with the rapid development of deep learning techniques in the realm of natural language processing, we have elected to use the patent-specific BERT-for-Patents model as our baseline model. This model has proven effective in numerous language tasks and, as such, serves as an appropriate standard baseline of comparison for patent text-based categorization tasks.

While we initially experimented with recent optimization methodologies related to BERT to enhance our baseline performance, our endeavors in this direction did not yield the improvements we had hoped for. The results indicated that these methods might not be ideally suited for our specific task, thus emphasizing the need for a more tailored approach.

Motivated by these initial findings, we propose an innovative model, named RGANN, which utilizes the syntactic graph structure to refine the embedding representation of the BERT-for-Patents model. This model leverages the inherent graph structure within patent texts to extract more nuanced features, thereby offering a more effective approach to patent multi-label classification.

Our empirical evaluations reveal that the RGANN model yields a significant improvement over the baseline BERT-for-Patents model, with an increase of 3.01% in F1-score on the Patent2M dataset. More impressively, on the larger and more challenging Patent14K dataset, the RGANN model registers an even larger improvement of 4.22% in F1-score. These substantial improvements over the baseline model underscore the efficacy of our proposed RGANN model, and open up new avenues for future research in patent multi-label classification.

B. Discussion

Our work does not fine-tune the entire BERT-for-Patents model. However, there is much evidence that fine-tuning the entire BERT model can lead to downstream task better performance. The computational resources required to fine-tune the whole BERT are large, so one way can be tried to use adapters to effective parameter fine-tuning, like AdapterBias [26], LLM-Adapters [27], GlossBERT [28], Parameter-Efficient learning [29]. Secondly, it is important how to improve the inference speed of BERT-related models. We leave these studies to the future.

Large language model (LLM) is a trend in the future. Now the large language models are not very good at classification tasks, such as the multi-label classification of patents. We hope that our datasets and baseline will further advance the Artificial General Intelligence (AGI) capabilities of large language model.

C. Limitation and Future Direction

Based on the above findings, three directions for further research are identified.

First, this work highlights the importance of syntactic dependency information in multi-label classification tasks. However, the impact of syntactic structures on other Natural Language Processing (NLP) tasks still requires further investigation. Although Shu [30] has explored the influence of parsing on certain tasks, the parsing capability of Large Language Models (LLMs) remains insufficiently studied. A deeper understanding of syntactic information across different NLP tasks may further enhance model performance and robustness.

Second, inference efficiency remains an important issue. Because of the high computational complexity of LLMs, inference time can be excessively long, which limits their

practical deployment. Although previous studies such as Mitra [31] and Tay [32] have attempted to reduce inference latency, transformer-based models still face challenges in achieving efficient inference. Therefore, developing more effective strategies for accelerating inference is an important direction for future research.

Third, the computational cost of fine-tuning the BERT-for-Patents model remains high. The large resource requirements highlight the need for more efficient fine-tuning strategies. One possible solution is the use of adapters, which introduce lightweight task-specific parameters without modifying the original model structure. This approach may help balance model performance and computational efficiency, though further research is still needed.

REFERENCES

[1] B.Degroote, P.Held, Analysis of the patent documentation coverage of the CPC in comparison with the IPC with a focus on Asian documentation, World Pat. Inf. 54 (2018) S78–S84. doi:10.1016/j.wpi.2017.10.001.

[2] Piroi, Florina, et al. "CLEF-IP 2011: Retrieval in the Intellectual Property Domain." CLEF (notebook papers/labs/workshop). 2011.

[3] Phillips v. AWH Corp., 415 F.3d 1303, 1312 (Fed. Cir. 2005) (en banc), (n.d.). https://casetext.com/case/phillips-v-awh-corp3.

[4] Radford, Alec, et al. "Improving language understanding by generative pre-training." (2018).

[5] J.Devlin, M.-W.Chang, K.Lee, K.Toutanova, BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding, in: Proc. 2019 Conf. North Am. Chapter Assoc. Comput. Linguist. Hum. Lang. Technol. NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Vol. 1 (Long Short Pap., 2019: pp. 4171–4186.

[6] Yang, Zhilin, et al. "Xlnet: Generalized autoregressive pretraining for language understanding." Advances in neural information processing systems 32 (2019).

[7] Brown, Tom, et al. "Language models are few-shot learners." Advances in neural information processing systems 33 (2020): 1877-1901.

[8] Li, Bohan, et al. "On the sentence embeddings from pre-trained language models." arXiv preprint arXiv:2011.05864 (2020).

[9] Su, Jianlin, et al. "Whitening sentence representations for better semantics and faster retrieval." arXiv preprint arXiv:2103.15316 (2021).

[10] Tianyu Gao, Xingcheng Yao, and Danqi Chen. 2021. SimCSE: Simple Contrastive Learning of Sentence Embeddings. In Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, pages 6894–6910, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

[11] Grawe M F, Martins C A, Bonfante A G. Automated patent classification using word embedding[C]//2017 16th IEEE International Conference on Machine Learning and Applications (ICMLA). IEEE, 2017: 408-411.

[12] Li S, Hu J, Cui Y, et al. DeepPatent: patent classification with convolutional neural networks and word embedding[J]. Scientometrics, 2018, 117: 721-744.

[13] Yu J, Huang L, Hu Y, et al. A structured representation framework for TRIZ-based Chinese Patent Classification via reinforcement learning[C]//2020 3rd International Conference on Artificial Intelligence and Big Data (ICAIBD). IEEE, 2020: 6-10.

[14] Shalaby M, Stutzki J, Schubert M, et al. An lstm approach to patent classification based on fixed hierarchy vectors[C]//Proceedings of the 2018 SIAM International Conference on Data Mining. Society for Industrial and Applied Mathematics, 2018: 495-503.

[15] Lee J S, Hsiang J. Patent classification by fine-tuning BERT language model[J]. World Patent Information, 2020, 61: 101965.

[16] OpenAI. "GPT-4 Technical Report." ArXiv abs/2303.08774 (2023): n. pag.

[17] Jiao, Wenxiang, et al. "Is ChatGPT a good translator? A preliminary study." arXiv preprint arXiv:2301.08745 (2023).

[18] Hendy, Amr, et al. "How good are gpt models at machine translation? a comprehensive evaluation." arXiv preprint arXiv:2302.09210 (2023).

[19] "Data Source Tables." Dimensions on Google BigQuery documentation. Accessed 29 Apr 2022. https://docs.dimensions.ai/bigquery/data-sources.html.

[20] Srebrovic, Rob, and Jay Yonamine. Leveraging the BERT algorithm for Patents with TensorFlow and BigQuery. Technical Report. Global Patents, Google https://services. google. com/fh/files/blogs/bert_for_patents_white_paper. pdf, 2020.

[21] Kingma, Durk P., and Prafulla Dhariwal. "Glow: Generative flow with invertible 1x1 convolutions." Advances in neural information processing systems 31 (2018).

[22] Shlens, Jonathon. "A tutorial on principal component analysis." arXiv preprint arXiv:1404.1100 (2014).

[23] Xu, Yinchuan, and Junlin Yang. "Look again at the syntax: Relational graph convolutional network for gendered ambiguous pronoun resolution." arXiv preprint arXiv:1905.08868 (2019).

[24] Cen, Yukuo, et al. "Representation learning for attributed multiplex heterogeneous network." Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining. 2019.

[25] Hamilton, Will, Zhitao Ying, and Jure Leskovec. "Inductive representation learning on large graphs." Advances in neural information processing systems 30 (2017).

[26] Fu, Chin-Lun, et al. "AdapterBias: Parameter-efficient Token-dependent Representation Shift for Adapters in NLP Tasks." arXiv preprint arXiv:2205.00305 (2022).

[27] Hu, Zhiqiang, et al. "LLM-Adapters: An Adapter Family for Parameter-Efficient Fine-Tuning of Large Language Models." arXiv preprint arXiv:2304.01933 (2023).

[28] Luyao Huang, Chi Sun, Xipeng Qiu, and Xuanjing Huang. 2019. GlossBERT: BERT for Word Sense Disambiguation with Gloss Knowledge. In Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP), pages 3509–3514, Hong Kong, China. Association for Computational Linguistics.

[29] Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, Sylvain Gelly Proceedings of the 36th International Conference on Machine Learning, PMLR 97:2790-2799, 2019.

[30] Raphael Shu, Hideki Nakayama, and Kyunghyun Cho. 2019. Generating Diverse Translations with Sentence Codes. In Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, pages 1823–1827, Florence, Italy. Association for Computational Linguistics.

[31] Mitra Mohtarami, James Glass, and Preslav Nakov. 2019. Contrastive Language Adaptation for Cross-Lingual Stance Detection. In Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP), pages 4442–4452, Hong Kong, China. Association for Computational Linguistics.

[32] Yi Tay, Dara Bahri, Liu Yang, Donald Metzler, Da-Cheng Juan Proceedings of the 37th International Conference on Machine Learning, PMLR 119:9438-9447, 2020.