Hierarchical Multilabel Patent Classification

Authors: (names, email and affiliations)

Xiaonan Hu: xh61@duke.edu

Xiaohan Wang: xw176@duke.edu

Allen Lin: sl604@duke.edu

Zilin Yin: zy108@duke.edu

Boyu Deng: bd132@duke.edu

Abstract:

Hierarchical multi-label classification (HMC) of patents is the task of classifying multiple labels for a patent, where each label is part of an underlying hierarchy of categories. With the increasing amount of patents shown up from different fields, and the failures of traditional multi-class classification approaches, the need for more robust automatic HMC patent classification methods increases. To build a robust classification system on this patent HMC task, we have tried several state-of-the-art classification models, in which some are commonly adopted in general classification problems, and others are specifically designed for hierarchical classification tasks. The models we have tried now include Bidirectional Encoder Representations from Transformers (BERT) multilabel classification method, Capsule Network Hierarchical Text Classification method, Hierarchical Attention-based Recurrent Model (HARNN) and a Recurrent Convolutional Neural Network (RCNN) Model. For now, we find that the BERT model achieves the best result on our patent dataset, with 0.28 micro F1 and 0.20 macro F1.

1. Introduction

With the development of novel studies in various fields, the number of current patents is increasing. This increasing amount of available digital patents description instantly calls for new and more robust automatic patent classification methods, more specifically, hierarchical multi-label classification approaches. However, several difficulties exist in this area. For example, patents, with numerous information, are presented in different languages, and some patents can be categorized into two or more related sections, classes, and subclasses in a hierarchical way. Therefore, HMC for patents is still a very challenging problem.

Even though plenty of researches have been done in the field of automated patent classification, very few of them were trained and tested using a multilingual dataset that resembles the size of the dataset given for our project. More importantly, most of the automated patent classification models do not handle hierarchical classification.

However, even though there are rarely any models or papers that we could directly apply to our project, by combining the gists from numerous papers and materials learned from class, we implemented several state-of-the-art patent hierarchical classification models, including Encoder Representations from Transformers (BERT) multilabel classification method, Capsule Network Hierarchical Text Classification method, Hierarchical Attention-based Recurrent Model (HARNN) and a Recurrent Convolutional Neural Network (RCNN) Model.

Along with the use of thoroughly designed data preprocessing and representation using special word embeddings,the best result was obtained by using the Bert model, with 0.28 micro F1 and 0.20 macro F1.

2. Related Work

The difficulty of the classification can be developed into three aspects. First, the amount of samples that need to be assigned is enormous. The patents dataset is large and contains 366,000 patents, 4,287,596,100 words and 815 classes in total. Second, finding an effective representation of word embedding is complex since each patent provides a bunch of respectively professional and technical concepts, between which a relationship is hard to discover. Also, some words may contain contrary meanings in different academic fields, which even more enlarges the hardship. Last, the words are shown in three languages, English, French and Germany and separated into three datasets, title, abstract and description. The complexity of this multilinguistic and multi-component dataset sort increases the challenge.

Researchers have done related work in the classification of patents. Zhu, H et al. adopted a symmetric hierarchical Convolutional Neural Network (CNN) method in the automatic classification of patents, where a high F1 score ranging around 80 was achieved. Xu, C et al. studied the label distribution learning in hierarchical classification and improved the algorithm. Hu, J et al. focused on the feature extraction of patents keyword and finished a comprehensive comparison between distributed representation based algorithms and other mainstream feature extraction methods. Tecent has published a github repository which has state of the art models on hierarchical english patient classification. As for the large scale hierarchical classification method, Wang, X. L et al. performed a meta-top-down method in datasets including IPC dataset from World Intellectual Property Organization and proved its accuracy. Pappas, N et al. performed and confirmed the efficiency of Hierarchical Attention Network applied in multilingual text classification tasks. Risch, J et al. promoted a domain-specific pre-trained word embedding for the patent document and probed its weakness as well as advantages. Lim, S et al. utilized the characteristics of the patent words rather than the data to conduct a hierarchical multilabel classification. Mao, Y et al applied deep reinforcement learning in the label assignment policy and incorporated different neural encoders as base models for

end-for-end learning in the task of hierarchical text classification. Shah, S et al. adopted Named Entity Recognition and subclassification in a multilingual language Identification and reached a weighted F-measure as 0.8082 for the target task.

3. Data

3.1 Data Overview

Collected from the European Patent Office under the Creative Commons Attribution 4.0 International Public license, the given patents dataset is combined of multilingual and multi label texts. The hierarchical structure of the dataset can be divided into three levels. A section level numbered with letters ranging from A to H gives a broad and conclusive overview of each patent, including chemistry, physics and others. Under the section level there is a class level group, numbered with two digit code ranging from 01 to 99, which gives a more specific description of each class, for example, B01, physical or chemical processes or apparatus in general. A subclass level lies beneath the class level and is numbered with letters from A to Z. The subclass level makes a precise description of the patent in each subclass, for example, A01C, planting; sowing; fertilising. All the patents content is presented in English, French and Germany.

Each sample patent in our dataset has four components to it: title, abstract, description and labels. The title is given in all three languages (English, French and Germany) but both the abstract and the description are only given in one of the three languages.

3.2 Preprocessing

For each sample in our dataset, we performed necessary preprocessing on the text data including stemming, stop words removal and, most importantly, common words elimination (we excluded words that frequently occurred in many patent class descriptions as they do not contribute to the uniqueness of individual sub-category).

We also decided to split the dataset into three sub datasets, an English subset, a French subset, and a Germany subset, based on the language used for the abstract and the description. For each sample in the dataset, we would check which language is being used for abstract and description, then remove the other two languages in the title and concatenate all four components to form a long string. This string is then assigned as the value of sample id, key, to be stored in a dictionary structure.

4. Methods

4.1 BERT

Motivation

BERT [17] has been successfully applied to a bunch of NLP tasks and all achieved state-of-the-art performance since it occured in 2018. One of the advantages of BERT is that it has plenty of pre-trained models, including even multilingual one. The pretrained-models are trained on Wikipedia, which is suitable for our patent classification task. Thus it's convenient for us to adopt the pre-trained multilingual model and then fine-tune it by our multilingual patent dataset. It's interesting to see whether BERT performs well on our HMC task.

Model

We utilized a *bert-based-multilingual-cased* pretrained model provided in github Transformer repo (https://github.com/huggingface/transformers). The model has 12-layer, 768-hidden, 12-heads, and 110M parameters. It was trained on cased text in the top 104 languages with the largest Wikipedias.

Traditional multi label classification model employs sigmoid on output, and then computes binary cross entropy (BCE) loss on each class. However, in our dataset, there are more than 500 classes, while most patents have only about 2-3 labels, thus a sigmoid-BCE procedure tends to lead to all 0 predictions. Therefore, in our experiments in this part, we first set the groundtruth of each patent label to be the 0/1 representation over the number of classes this patent belongs to, so that the generated groundtruth vector adds up to one. For example, if a patent previously has a label (1, 1, 0, 1, 0), then we set the label to (⅓, ⅓, 0, ⅓, 0). Then we adopt the Softmax layer on output, and use cross entropy loss.

Experiment

Since a multilingual pretrained model is adopted, we don't put any language constraint on the input text. However, the pretrained model has a limit for the sentence of input that its length should be no larger than 512, so only the titles and abstract of each patent are utilized. Our batch size is 16, learning rate is 2e-5, weight decay is 0.01.

Softmax layer only outputs a score for each class for each patent, and we still have to transform the score into 0/1 labels. To this end, we first employed a grid search to find the best threshold that has the highest F1 score. This threshold is then utilized to to binary the final labels.

Our Bert model achieves 0.28 micro F1 and 0.20 macro F1. One of the reasons that BERT doesn't perform well on this task should be that there are too many classes, and no hierarchical information is used when fine tuning the model. This experiment illustrates the importance of hierarchical information in our HMC task.

4.2 HARNN Model

Motivation

For this hierarchical multi-class classification task, we also apply Hierarchical Attention-based Recurrent Model (Huang et. al.) to classify patent data with natural hierarchical structure. Combining hierarchical loss function design, Attention-mechanism and Bi-LSTM, this framework has huge potential in capturing the semantic information of each layer of patent representation and passing the learnt information to the next layer.

Data Preparation

For data preparation, we first recognize and divide the patent data into English-, French- and German-based patents datasets. Word embeddings were trained separately by the skip-gram model using patent datasets from the tree datasets. Within each language category, for each patent, we process its title by deleting the stop words, and shorten the patent data by picking informative words in the abstract and the description with predefined TF-iDF scores cutoff threshold.

Model Description

After obtaining a list of words in the preprocessing phase, each patent's document representation is constructed first through the embedding layer which encodes each word using the pre-trained word embeddings. It is then followed by a Bi-LSTM layer which learns a hidden representation for each word within a patent document, encoding the before and after contexts of each given word. It then applies average pooling to the hidden embeddings into a single vector which has length twice as the hidden vector. This vector will be used as the patent representation.

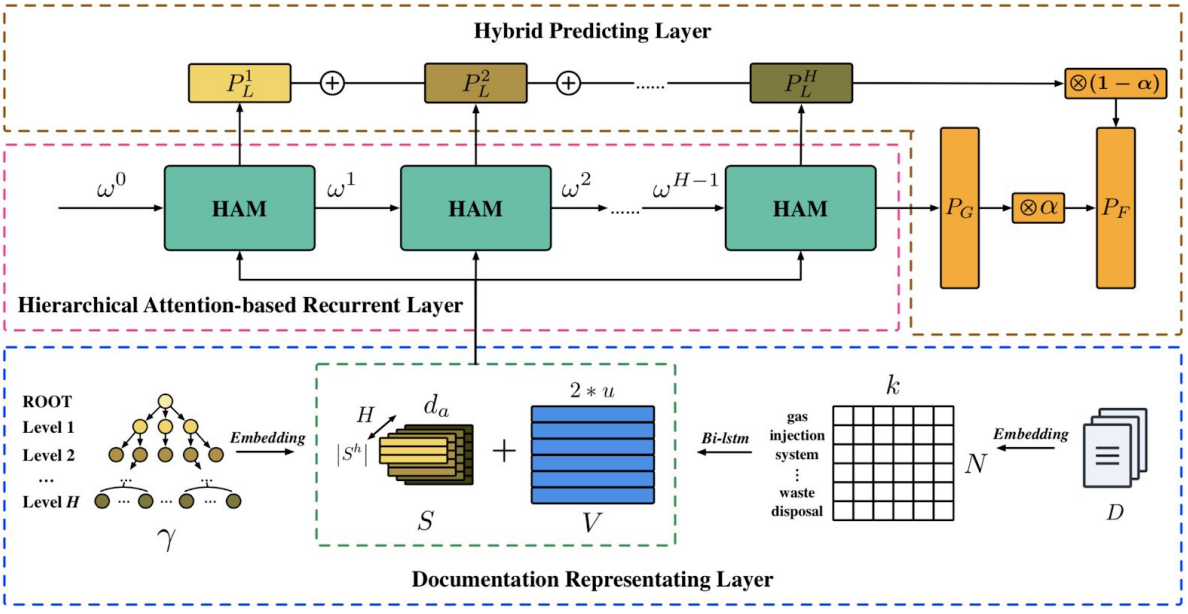


Figure :The HARNN framework takes a given document text D and the corresponding hierarchical category structure γ as inputs, and outputs the hierarchical categories L for the given document. Adapted from "Hierarchical Multi-label Text Classification: An Attention-based Recurrent Network Approach" by Huang, Wei, Enhong Chen, Qi Liu, Yuying Chen, Zai Huang, Yang Liu, Zhou Zhao, Dan Zhang, and Shijin Wang. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, pp. 1051-1060. 2019.

After obtaining the representation, a Hierarchical Attention-based Recurrent (HARL) Layer is used to learn the categories at each hierarchical level in a top-down fashion. Each level of HARL consists of a Hierarchical Attention Memory Unit which consists of three components: Text Category Attention (TCA), Class Prediction Module (CPM), Class Dependency Module (CDM). TCA captures the relations between the text embeddings and the label at the current level. CPM generates the prediction of category at the current level and learns a unified representation with the information passed from the previous level. Finally, CDM is constructed to depict the hierarchical dependency of classes across different levels.

Following HARL, a Hybrid Prediction Layer outputs all the labels related to a patent for all levels within the hierarchy. It incorporates the results from both the local prediction and the global prediction with equal weighting.

Further Improvements

The current F1 score, around 0.20, is unsatisfiable when HARNN is tested on the test data which, and further improvements are needed. We will continue investigating ways to improve the model. First of all, we suspect model overfitting and We still will try to stop training earlier. Second, many testing patents have labels that do not appear in the training set. We need to find some generic representation which can overcome this issue.

4.3 Capsule Model

4.3.1 **Motivation**

For hierarchical multilabel classification (HMC) tasks, traditional approaches fail to generalize the classification processes. For example, for a simple multilabel classification task, only one model is needed to accomplish the task. However, for the case of hierarchical multilabel classification, if used traditional models, multiple models may have to be built and set up to work together, having each one of them covering a certain part of the hierarchy in order to accomplish the complicated task. This could result in significant computation cost along with extensive implementation difficulty. Hence, more complicated, robust and integrated models are needed.

4.3.2 **Data Preprocessing**

Spacy is used for data preprocessing (e.g. tokenization, lemmatization) for the Capsule model as it is considered to be fast in preprocessing the dataset. Also, wiki's word embeddings is used by the model. The data has also been manually cleaned, parsed to better fit into the predefined structure of the Capsule model.

4.3.3 **Model**

We used the *Hierarchical multi-label text classification Capsule* model in github Transformer reo (https://github.com/uhh-lt/BlurbGenreCollection-HMC). Capsule model is a global approach model. That means it uses one classifier to capture the

entire hierarchy at once, without having multiple classifiers to take care of different parts of the hierarchy. The main idea of a capsule network is to have several neuron groups, with each group taking care of a certain part of the hierarchy. A graphical representation of the architecture of a Capsule network is as below:

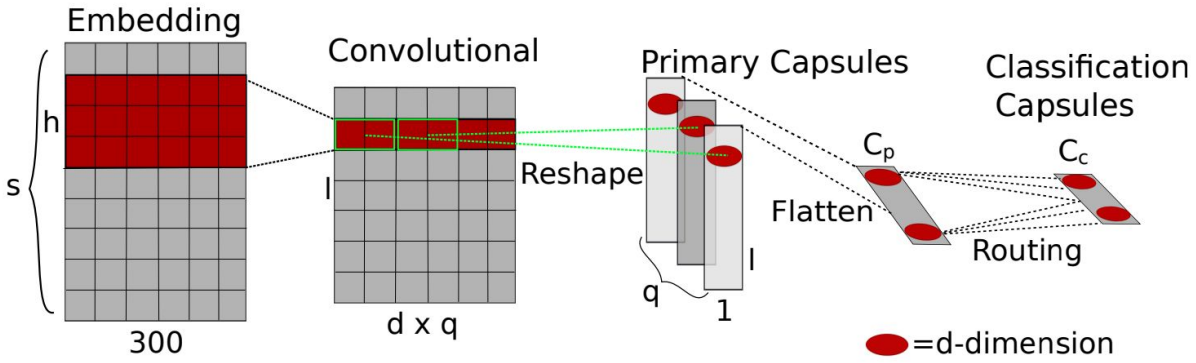


Figure ??. Architecture of capsule network with d being the dimensionality of a capsule's output (Aly et al., 2019)

As shown in the figure above, the left half part is very similar to what a convolutional neural network (CNN) does. Primary capsules are essentially the input of capsules in the first capsule layer of a capsule network. Basically the input of the primary capsules would be a CNN layer that has gone through several common CNN processes like filtering and pool (the specific details of the processes could be adjusted). And the output of the primary capsules would be a series of vectors representing the latent features of words (Xiao et al., 2018). Then for each capsule j, which is called classification capsule in the next layer in the network, it uses the routing algorithm to cluster the outputs. In other words, the output of a classification capsule would be a vector of the probability of the corresponding category that that capsule is responsible for. The specific procedure of routing algorithm is as below:

Procedure 1 Routing algorithm.

1: **procedure** ROUTING($\hat{\boldsymbol{u}}_{j|i}, r, l$)
2: for all capsule i in layer l and capsule j in layer $(l+1)$: $b_{ij} \leftarrow 0$.
3: **for** r iterations **do**
4: for all capsule i in layer l: $\mathbf{c}_i \leftarrow \texttt{softmax}(\mathbf{b}_i)$
5: for all capsule j in layer $(l+1)$: $\mathbf{s}_j \leftarrow \sum_i c_{ij}\hat{\mathbf{u}}_{j|i}$
6: for all capsule j in layer $(l+1)$: $\mathbf{v}_j \leftarrow \texttt{squash}(\mathbf{s}_j)$
7: for all capsule i in layer l and capsule j in layer $(l+1)$: $b_{ij} \leftarrow b_{ij} + \hat{\mathbf{u}}_{j|i}.\mathbf{v}_j$
return $\mathbf{v}_j$

Figure ??. Routing Algorithm (Sabour et al., 2017)

With reference to figure ?? above, some important terms shall be explained. vj denotes the vector output of capsule j, while sj denotes the total output. cij denotes the coupling coefficients that are determined by the iterative dynamic routing

process, and lastly, bij are the log prior probabilities that capsule i should be coupled to capsule j (Sabour et al., 2017).

For the set up of the Capsule model, some unique settings that are considered important to the capsule model used for the study are as follow: the dimensionality of capsules on the final layer is 16; number of capsules per feature map is 50. Besides, general model options like batch size, activation threshold, etc. are also tuned and well set for optimal model classification performance.

4.3.4 **Experiment**

As all other models, F1 micro and macro are set as the scoring metrics for the Capsule model. Due to hardware limitation, we weren't able to put all the data into the model. Instead, a subset of the original dataset is taken out and been put into the Capsule model, resulting in an F1 micro score of 0.25 and F1 macro score of 0.25 on the test set. As mentioned before, the scores are based only on a subset of the data, so that they are not representative or comprehensive enough.

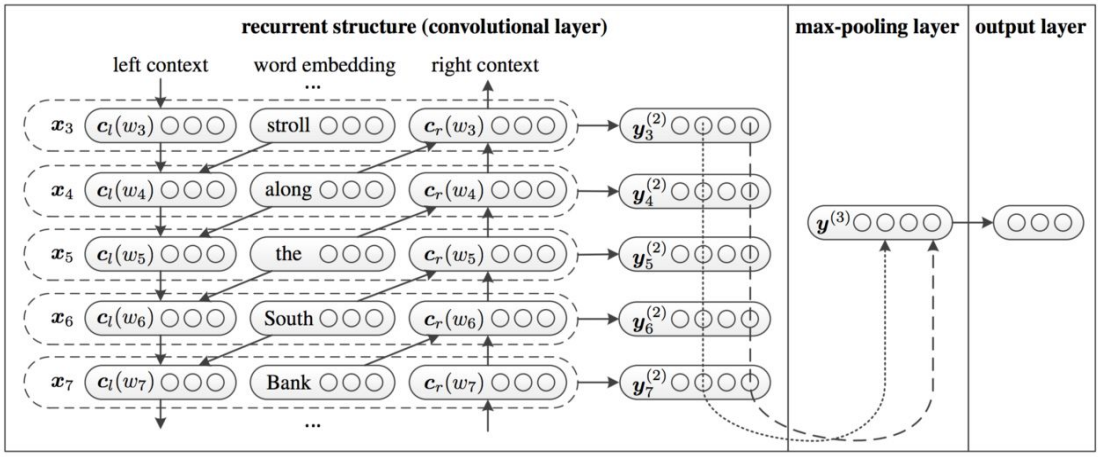
4.4 RCNN Model

4.4.1 Motivation

For text classification tasks, both recurrent neural networks(RNN) and convolutional neural networks(CNN) suffer some drawbacks. The advantage of RNN is the ability to better capture the contextual information; however, RNN is a biased model and could reduce the effectiveness when it is used to capture the semantics of a whole document. While handling the bias problem of RNN, previous CNNs tend to use simple convolutional kernels such as a fixed window When using such kernels, it is difficult to determine the window size: small window sizes may result in the loss of some critical information, whereas large windows result in an enormous parameter space. [19]

Found on github, a Hierarchical Multi-label Text Classification toolkit [20] provided by Tencent, RCNN achieved the highest accuracy when dealing with hierarchical classification tasks. Therefore we decided to modify the model and evaluate its performance on your dataset.

4.4.2 Model



Equations

$$c_l(w_i) = f(W^{(l)}c_l(w_{i-1}) + W^{(sl)}e(w_{i-1})) \quad (1)$$

$$c_r(w_i) = f(W^{(r)}c_r(w_{i+1}) + W^{(sr)}e(w_{i+1})) \quad (2)$$

- $(1), (2)$ show how to compute context vectors $c_l(w_i), c_r(w_i)$. Note that the equations are slightly different from vanilla RNN:

$$h_t = f(W_{xh}x_t + W_{hh}h_{t-1})$$

$$x_i = [c_l(w_i); e(w_i); c_r(w_i)] \quad (3)$$

- (3) shows how to represent a word w_i.

$$y_i^{(2)} = tanh\left(W^{(2)}x_i + b^{(2)}\right) \quad (4)$$

- (4) shows how to compute latent semantic vector $y_i^{(2)}$, in which semantic factor will be analyzed to the most useful factor for representing the text.

Implementation Details:

Improving upon the original model, we used BiLSTM instead of vanilla RNN as proposed in the paper. For parameters, we used embedding size of 300, batch size of 256, dropout rate of .7 and hidden size of 128. We used SGD as optimizer, Negative Log Likelihood as our loss function, and pre-trained Glove Embeddings for constructing word vectors.

4.4.3 Experiment

The current scores for our RCNN model is not ideal. both the macro and micro F1 score are around 0.1.This could happen due to the lack of a more effective embedding. Since our dataset is much bigger than the dataset used by Tencent to

evaluate their RCNN. We will keep working on making a better and more effective embedding.

5. CONCLUSIONS AND OPEN PROBLEMS (on final report)

In this report, we tried BERT, Capsule Network, HARNN and a RCNN Model to tackle the HMC problem, and obtained some decent results with the BERT model performing the best. Hierarchical Multiclass classification applied in the context of patent classification is innately hard and is far from being solved. The current best performance in the competition has less than 55% F-1 score, which is pretty high considering that there are more than 600 classes and each patent can be assigned to multiple classes. Further investigations, such as more effective embedding, are needed to improve our current models.

Our data is available at:

https://competitions.codalab.org/competitions/22533#learn_the_details-data-description

Our code is available at:

https://github.com/XiaonanHu/Hierarchical_multilabel_classification

6. REFERENCES

1. Hu, J., Li, S., Hu, J., & Yang, G. (2018). A Hierarchical Feature Extraction Model for Multi-Label Mechanical Patent Classification. *Sustainability*, *10*(1), 219.
2. Wang, X. L., Zhao, H., & Lu, B. L. (2013). A meta-top-down method for large-scale hierarchical classification. *IEEE Transactions on Knowledge and Data Engineering*, *26*(3), 500-513.
3. Zhu, H., He, C., Fang, Y., Ge, B., Xing, M., & Xiao, W. (2020). Patent Automatic Classification Based on Symmetric Hierarchical Convolutional Neural Network. *Symmetry*, *12*(2), 186.
4. Risch, J., & Krestel, R. (2019). Domain-specific word embeddings for patent classification. *Data Technologies and Applications*.
5. Xu, C., & Geng, X. (2019, July). Hierarchical classification based on label distribution learning. In *Proceedings of the AAAI Conference on Artificial Intelligence* (Vol. 33, pp. 5533-5540).
6. Mao, Y., Tian, J., Han, J., & Ren, X. (2019). Hierarchical Text Classification with Reinforced Label Assignment. *arXiv preprint arXiv:1908.10419*.
7. Hu, J., Li, S., Yao, Y., Yu, L., Yang, G., & Hu, J. (2018). Patent keyword extraction algorithm based on distributed representation for patent classification. *Entropy*, *20*(2), 104.

8. Lim, S., & Kwon, Y. (2016). IPC multi-label classification applying the characteristics of patent documents. In *Advances in Computer Science and Ubiquitous Computing* (pp. 166-172). Springer, Singapore.
9. Gomez, J. C., & Moens, M. F. (2014). A survey of automated hierarchical classification of patents. In *Professional search in the modern world* (pp. 215-249). Springer, Cham.
10. Pappas, N., & Popescu-Belis, A. (2017). Multilingual hierarchical attention networks for document classification. *arXiv preprint arXiv:1707.00896*.
11. Shah, S., Jain, V., Jain, S., Mittal, A., Verma, J., Tripathi, S., & Kumar, R. (2015). Hierarchical classification for Multilingual Language Identification and Named Entity Recognition. In *FIRE Workshops* (pp. 33-36).
12. Protasiewicz, J., Stanislawek, T., & Dadas, S. (2015, October). Multilingual and hierarchical classification of large datasets of scientific publications. In *2015 IEEE International Conference on Systems, Man, and Cybernetics* (pp. 1670-1675). IEEE.
13. Jacquet, G., Piskorski, J., & Chesney, S. (2019, April). Out-of-context fine-grained multi-word entity classification: exploring token, character n-gram and NN-based models for multilingual entity classification. In *Proceedings of the 34th ACM/SIGAPP Symposium on Applied Computing* (pp. 1001-1010).
14. Helmy, A. A. (2019). A multilingual encoding method for text classification and dialect identification using convolutional neural network. *arXiv preprint arXiv:1903.07588*.
15. Hall, J. W. (2017). Examination of machine learning methods for multi-label classification of intellectual property documents.
16. Lifang, Y., Sijun, Q., & Huan, Z. (2017, April). Feature selection algorithm for hierarchical text classification using Kullback-Leibler divergence. In *2017 IEEE 2nd International Conference on Cloud Computing and Big Data Analysis (ICCCBDA)* (pp. 421-424). IEEE.
17. Devlin J, Chang M W, Lee K, et al. Bert: Pre-training of deep bidirectional transformers for language understanding[J]. arXiv preprint arXiv:1810.04805, 2018.
18. Huang, Wei, Enhong Chen, Qi Liu, Yuying Chen, Zai Huang, Yang Liu, Zhou Zhao, Dan Zhang, and Shijin Wang. "Hierarchical Multi-label Text Classification: An Attention-based Recurrent Network Approach." In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, pp. 1051-1060. 2019.
19. Lai, Siwei et al. “Recurrent Convolutional Neural Networks for Text Classification.” *AAAI* (2015).
20. Tencent. “Tencent/NeuralNLP-NeuralClassifier.” *GitHub*, 9 Apr. 2020, github.com/Tencent/NeuralNLP-NeuralClassifier.
21. Sara Sabour, Nicholas Frosst, and Geoffrey E. Hinton. 2017. Dynamic routing between capsules. In Advances in Neural Information Processing Systems 30, pages 3856–3866, Long Beach, CA, USA.

22. Liqiang Xiao, Honglun Zhang, Wenqing Chen, Yongkun Wang, and Yaohui Jin. 2018. MCapsNet: Capsule network for text with multi-task learning. In Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, page 4565–4574, Brussels, Belgium.