A Multi-Task Approach to Neural Multi-Label Hierarchical Patent Classification using Transformers [⋆]

Subhash Chandra Pujari[1,2], Annemarie Friedrich[1], and Jannik Strötgen[1]

[1] Bosch Center for Artificial Intelligence, Renningen, Germany
[2] Institute of Computer Science, Heidelberg University, Heidelberg, Germany
subhashchandra.pujari@de.bosch.com
annemarie.friedrich@de.bosch.com
jannik.stroetgen@de.bosch.com

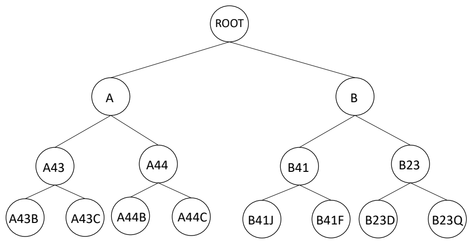
Abstract. With the aim of facilitating internal processes as well as search applications, patent offices categorize documents into taxonomies such as the Cooperative Patent Categorization. This task corresponds to a multi-label hierarchical text classification problem. Recent approaches based on pre-trained neural language models have shown promising performance by focusing on leaf-level label prediction. Prior works using intrinsically hierarchical algorithms, which learn a separate classifier for each node in the hierarchy, have also demonstrated their effectiveness despite being based on symbolic feature inventories. However, training one transformer-based classifier per node is computationally infeasible due to memory constraints. In this work, we propose a *Transformer-based Multi-task Model* (TMM) overcoming this limitation. Using a multi-task setup and sharing a single underlying language model, we train one classifier per node. To the best of our knowledge, our work constitutes the first approach to patent classification combining transformers and hierarchical algorithms. We outperform several non-neural and neural baselines on the WIPO-alpha dataset as well as on a new dataset of 70k patents, which we publish along with this work. Our analysis reveals that our approach achieves much higher recall while keeping precision high. Strong increases on macro-average scores demonstrate that our model also performs much better for infrequent labels. An extended version of the model with additional connections reflecting the label taxonomy results in a further increase of recall especially at the lower levels of the hierarchy.

Keywords: patent classification · hierarchical classification · multi-label classification · neural modeling · multi-task learning.

1 Introduction

A patent is a legal text document describing an invention and granting its owner exclusive rights for monetary exploitation thereof. Upon submission of a patent application, patent offices assign one or several labels categorizing the described invention according to a taxonomy such as the Cooperative Patent Classification (CPC). This scheme,

[⋆] The final authenticated version is available online at https://doi.org/10.1007/978-3-030-72113-8_34



Level	ID	Description
Section	A	Human Necessities
Class	A43	Footwear
Subclass	A43B	Characteristic features of footwear; parts of footwear

Fig. 1: Excerpt of the hierarchical **Cooperative Patent Classification** (CPC) scheme.

developed jointly by the US Patent and Trademark Office (USPTO) and the European Patent Office, organizes types of inventions in a hierarchical tree structure as illustrated in Fig. 1. CPC information is used internally by the patent offices, e.g., for routing the application to the respective experts. It is also released publicly with each patent in order to facilitate search-related tasks including the retrieval and filtering of patents.

From a machine learning (ML) point of view, assigning CPC codes to patents constitutes a hierarchical multi-label text classification problem and has high relevance to a variety of information-retrieval (IR) related real-life tasks. First, with currently almost 2,000 patent applications being submitted per day to USPTO alone, the automatic prediction of CPC codes helps to speed up manual work considerably. Second, patent language often intentionally conceals the type of invention by avoiding terminology commonly used in technical reports [28]. Detecting the underlying CPC codes present in patents, scientific reports or other types of text-based queries will lead to more meaningful rankings of patents during retrieval. Finally, the CPC taxonomy itself is under constant development with new categories being added or parts being restructured. Accurate automatic classification methods will help to keep patent databases up-to-date with the taxonomy, a prerequisite for the above mentioned search applications.

At its top level, the CPC scheme has nine *sections*. *Subclasses* are further divided into *main groups* and *subgroups*, amounting to a total of 250,000 categories. Patent classification has been addressed by the IR and ML communities in the context of several shared tasks organized by ALTA and CLEF-IP [27,30], operating at various granularities of the taxonomy. In this paper, following previous work [21,22], we address the first three levels of the taxonomy, resulting in hierarchical multi-label classification tasks with huge label inventories of around 600 classes in our datasets (Sec. 4).

We address this large-scale classification task using a novel combination of a pre-trained language model [3,8] and a *local* hierarchical learning algorithm. Such algorithms train one "local" classifier per node of the taxonomy predicting whether an instance belongs to the respective category or not, and have been shown to be highly effective for hierarchical patent classification in previous work using symbolic features such as n-grams and part-of-speech tags [4,5]. Similarly, approaches based on contextual word embeddings and transformers have shown promising performance [12,21,22]. They apply a *flat* strategy, i.e., they train a single classifier that simply predicts leaf-level classes and infer ancestor classes from them. In this paper, we combine the advantages of using powerful document embeddings generated by a pre-trained language model with the gains that can be achieved by localizing decisions. It is arguably computationally infeasible in most infrastructures to instantiate hundreds of transformer-based language models in parallel. Therefore, we propose a new multi-task based neural architecture for hierarchical multi-label classification in which the individual classifiers corresponding to the nodes of the taxonomy constitute the classification heads in a neural network, sharing the same underlying transformer-based language model. In addition, we create a variant adding connections between the classification heads that are related in the label taxonomy.

We benchmark our approach with a variety of non-neural and neural hierarchical text classification algorithms using the WIPO-alpha dataset and a new patent dataset spanning the years 2006-2019. We publish the latter along with our paper. On both datasets, our models strongly outperform prior work both in terms of macro- and micro-averages. Our detailed analysis of performance at the different levels of the taxonomy reveals that our models are much better (a) at predicting less frequent categories and (b) at predicting finer-grained labels. Adding taxonomy-based connections to our model results in further increases in recall especially for leaf-level labels.

Our contributions are as follows. (i) We propose a novel *Transformer-based Multi-task Model* neural-network architecture and a variant adding hierarchical connections (Sec. 3). We open-source our implementation in order to foster future research. (ii) We sample a new dataset of 70k recent USPTO patents that we make publicly available for benchmarking (Sec. 4).[3] (iii) We perform an in-depth analysis demonstrating that our models strongly outperform prior work, achieving much better accuracy on the lower levels of the hierarchy as well as for less frequent CPC classes (Sec. 5).

2 Related Work

Despite having been studied in the data mining, ML, and IR communities for many years [2], text classification remains a very active research field addressing a variety of domains [15,23,37]. Since the seminal works using Convolutional Neural Networks (CNNs) for sentence classification [16,18], neural modeling has become the predominant approach. In this work, we focus on **hierarchical text classification** [34], in which the label set constitutes a hierarchy. While some architectures or algorithms directly reflect these taxonomies [4,5], others apply *flat* or *global* approaches either predicting only leaf-level labels or simply treating all labels independently [14,21,22].

We here address the task of **patent classification**, which while constituting a hierarchical multi-label text classification problem, is often addressed using flat classifiers [11], though with several notable exceptions [4,5,36]. Patent documents are usually represented by transforming the text of their title and abstract into a feature vector for classification. Recent work uses the CPC scheme explained above, while some older datasets use the International Patent Categorization (IPC), which is roughly speaking a predecessor of CPC. In a recent shared task on patent classification [27], an approach training separate SVM classifiers per node using simple n-gram and POS-tag based features [5] performed comparably to a flat neural approach [12] based on the ULMFiT contextual language model [13]. The work of Li et al. [22], based on [18] and

[3] https://github.com/boschresearch/hierarchical_patent_classification_ecir2021.

optimized by [1], proposes a convolutional neural network based on non-contextual word2vec [26] embeddings predicting IPC codes on subclass level. In this work, we compare to the state-of-the-art HARNN system [14] (see Sec. 5.3). HARNN generates document embeddings with a BiLSTM initialized using word2vec, feeds these through a hierarchical attention-based memory unit that learns different attention weights per category, and finally predicts categories by combining hidden *local* and *global* information. The former relates to level-wise predictions and the latter consists of predictions for the entire taxonomy. Further, neural work on patent classification [35] employs graph-convolutional networks using word embeddings inferred from a word-document co-occurrence graph. Hierarchical patent classification has also been addressed as a sequence generation problem using an attention-based neural network model [32].

Outside the context of patent classification, [40] uses a very similar approach to [35]; [20] and [38] address **neural hierarchical text classification** by training level-wise classifiers and chaining predictions top-down. Similar to our work, [29] propose a CNN model in which the hierarchy of labels is leveraged by regularizing the deep architecture with dependencies among labels. A weakly-supervised hierarchical classification approach is suggested in [24]. Given a few user-provided seeds, the system generates pseudo-documents that are used for bootstrapping a neural hierarchical classifier including an LSTM-based language model.

Recently, **transformer**-based **neural language models** such as BERT [8] have been shown to be highly effective for a variety of natural language processing tasks [33], following a "pre-train and fine-tune" approach. In the context of patent classification, PatentBERT [21] adds a single hidden layer on top of BERT, mapping the CLS embedding to a sigmoid output in order to predict CPC labels on subclass level. [17] employs the same idea, predicting relevance with regard to a pre-specified topic.

Our work differs from previous work in the area of hierarchical patent classification in the following aspects. First, instead of predicting labels at a single hierarchical level [1,5,21,22], we model predictions across the label taxonomy. Second, unlike the flat classification [21,22] model architectures, we make use of the taxonomy by transferring information from the parent label to child labels. Finally, to the best of our knowledge, our approach is the first to combine powerful transformer-based language models with an intrinsically hierarchical algorithm for patent classification.

3 Model Architecture

3.1 Overview

We propose a neural hierarchical classification architecture as illustrated in Fig. 2. We assume the label set $L = \{l_1, l_2, l_3, ..., l_n\}$ in which labels are arranged hierarchically. The task consists in assigning a subset of L to each input document. For each predicted label, the respective ancestors should also be contained in the output set.

We create distributed representations of the textual input using a pre-trained transformer-based neural language model. For each label in the label set, we train a binary classifier that decides whether an instance belongs to the respective category or not. The ensemble of classifiers is trained in a multi-task setup and makes use of a single

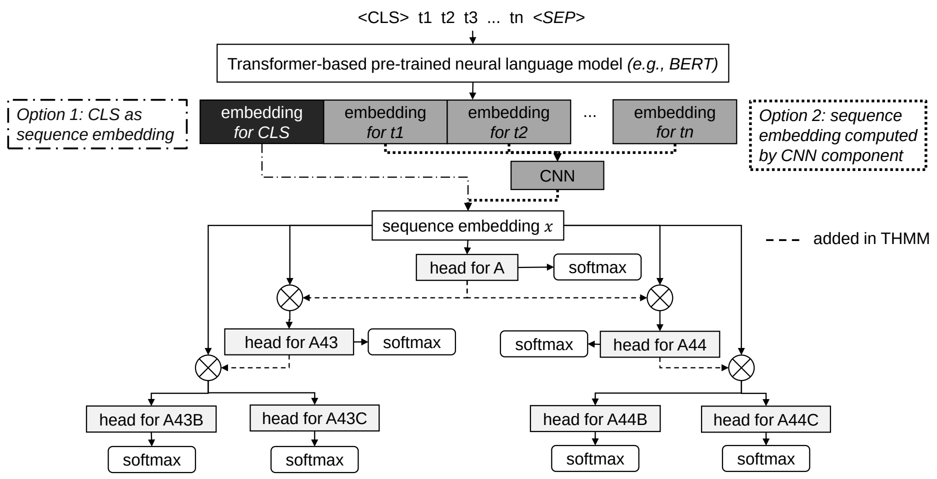


Fig. 2: Architecture of our **THMM** model for our running example from Fig. 1. The **TMM** version is the same but without the dashed connections between classification heads. Each classification head consists of three dense layers and predicts whether an instance belongs to the respective category or not.

underlying SciBERT neural language model [3] for creating document representations. SciBERT has been trained on a corpus of scientific publications and is hence closer to the patent domain than the standard BERT model [8]. In the terminology of multi-task learning, each of these classification heads addresses one *task*. Hence, each label-specific binary classifier constitutes a classification head in our multi-task based neural network architecture. In other words, our **Transformer-based Multi-task Model (TMM)** consists of a single transformer model with n heads where n corresponds to the number of labels in the hierarchy. Parameters of the transformer model (and of optional CNN layers) are shared in a hard way. In addition, each classification head has its own set of parameters. Further, to analyze the impact of sharing information between hierarchically related tasks, we propose an extended architecture which adds links between the network components corresponding to nodes that are linked in the hierarchy. We call this latter model **Transformer-based Hierarchical Multi-task Model (THMM)**.

3.2 Transformer-Based Language Model Based Document Representation

Similar to prior work on neural patent classification [14,21,22], we use the patent's title and abstract as input to our model. We concatenate them, word-piece tokenize the text and prepend the special CLS token. We leverage the transformer's output embeddings for the two variants of our model in the following ways: (i) We use the embedding generated for the CLS token (left-hand side path in Fig. 2), which can be regarded as capturing the semantics of the entire input text sequence [8]. However, the CLS token has been designed for next sentence prediction and it is unclear how effective its embedding is for representing long sequences as in our case. Hence, we also test a second option that explicitly considers the entire sequence: (ii) We compute a document embedding from the embeddings generated for each word-piece token by feeding them into a CNN (right/dotted path in Fig. 2). For details on the latter, see Sec. 5.2.

3.3 Classification Heads

We next detail the architecture of the classification heads. For the **Transformer-based Multi-task Model (TMM)**, we create an independent head for each label. The input for each head is the document embedding x corresponding to the embedding of the CLS token or the CNN's output. Each head consists of two dense layers, both with ReLU activation, followed by a two-dimensional dense output layer producing logits. Finally, we perform classification by means of a softmax operation.

In the **Transformer-based Hierarchical Multi-task Model (THMM)**, we add connections between the classification heads as specified by the label taxonomy. As in the TMM, each classification head computes the logits for the binary decision using two fully connected dense layers. However, in this case (see Equation (1)), the first hidden layer of the classification head for l_i additionally takes into account $h^2_{l_j}$, an output from the second (intermediate) dense layer of the head corresponding to l_i's parent l_j. It computes a hidden representation $h^1_{l_i}$ by performing a linear transformation on the concatenation ($\oplus$) of the sequence embedding x and $h^2_{l_j}$. If l_i does not have a parent in the taxonomy, the input to its classification head is simply x. The $parent(l_i, l_j)$ relation evaluates to $true$ if l_j is the parent of l_i, and to $false$ otherwise. ϕ is the ReLU activation function.

$$h^1_{l_i} = \begin{cases} \phi(W^1_{l_i}(h^2_{l_j} \oplus x) + b^1_{l_i}) & \text{if there is a } l_j \text{ with } parent(l_i, l_j) = true \\ \phi(W^1_{l_i}x + b^1_{l_i}) & \text{if } parent(l_i, ROOT) \end{cases} \tag{1}$$

$$h^2_{l_i} = \phi(W^2_{l_i}(h^1_{l_i}) + b^2_{l_i}) \qquad h^3_{l_i} = \phi(W^3_{l_i}(h^2_{l_i}) + b^3_{l_i}) \tag{2}$$

As in TMM, the hidden representation $h^1_{l_i}$ is passed through two further dense layers (Equation (2)) and mapped to a two-dimensional logit vector $h^3_{l_i}$. This serves as input to a softmax layer that performs the prediction whether label l_i applies to the instance. For training our models, we use binary cross entropy loss and weight all "tasks" equally.

4 Patent Classification Datasets

In this section, we give details on the two datasets we use for our experiments. To ensure comparability with prior work, we use WIPO-alpha, which contains 75k patents from 1998-2001 annotated with the IPC scheme. As domains, writing style and terminology of patents evolve over time, we also experiment with more recent data. We create a new dataset of 70k USPTO patents spanning the years 2006-2019, using the more recent CPC scheme. We release this dataset to ensure reproducibility of our study.

4.1 USPTO Dataset

We sample a dataset of 70k patents from the USPTO patents data dump[4] as follows. With the aim of creating a realistic setup in which models predict labels for newer patents based on older data, similarly to [9], we split the dataset temporally, assigning the documents from years 2006-2017 to the training set, 2018 to dev and 2019 to test. Our training sample contains 50k patents, and the dev and test sets 10k each.

We address label sparsity by up-sampling the least frequent labels by adding patents carrying the infrequent label such that each label occurs at least 10 times in the training set. Dev and test distributions are not changed. Fig. 3b shows that for some labels, there are many instances, but the distribution has a long tail. As shown in Figure 3a, the total number of labels at leaf node, i.e., subclass, level is 630 for train, 575 in dev, and 573 for test. There is one label occurring in dev that does not have any associated training instances. In the test split, there are 7 such labels. The average number of labels per patent is around 1.5 on the first level of the hierarchy and up to 2.32 on the leaf level, with the latter increasing from 1.8 in 2014 to 2.3 in 2019. This reflects a tendency towards more interdisciplinary inventions and further demonstrates the need to take the temporal dimension into account when training and evaluating models.

4.2 WIPO-alpha

The WIPO-alpha dataset[5] contains about 46k training instances and 29k test instances. The patent documents were published between 1998 and 2002, with test instances sampled randomly.[6] There are 602 labels in train and 576 test labels at subclass level. As there is no pre-existing split, we sample a validation (dev) set from train by selecting 20% of the data points at random. There are 22 labels with instances in test but without examples in the training data at subclass level. The IPC code in the dataset is defined using the seventh edition of IPC which labels each patent with a main IPC code and a set of secondary IPC codes. Unlike prior work [1], which considers only the main IPC code and benchmarks the models in a single-label flat classification setting, we consider all IPC codes in a hierarchical multi-label classification setting.

5 Experiments

In this section, we first describe our experimental setup including evaluation metrics, baselines and implementation details. We then discuss our experimental results in detail.

5.1 Evaluation Metrics

For evaluating our models, we use *hierarchical* precision, recall and F1-score as proposed by [19] and defined as $hP = \frac{\sum |P_i \cap T_i|}{\sum P_i}$ and $hR = \frac{\sum |P_i \cap T_i|}{\sum T_i}$. For each test instance i, the set P_i consists of all predicted labels and their respective ancestors. T_i

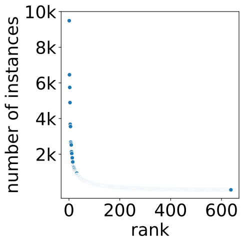
[4] https://www.patentsview.org/download

[5] https://www.wipo.int/classifications/ipc/en/ITsupport/Categorization/dataset/

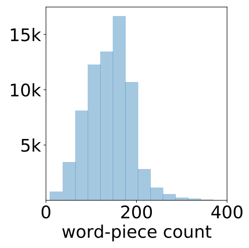
[6] See WIPO-alpha readme and personal correspondence with authors.

	splits	total labels			average labels per patent		
level		1	2	3	1	2	3
USPTO	train	9	128	630	1.49	1.69	1.98
	dev	9	126	575	1.56	1.84	2.25
	test	9	127	573	1.56	1.89	2.32
WIPO	train	8	123	602	1.22	1.34	1.49
	dev	8	120	544	1.22	1.35	1.49
	test	8	128	576	1.22	1.35	1.51

(a)



(b)



(c)

Fig. 3: **Corpus statistics** of USPTO and WIPO-alpha datasets. (a) Label counts by level. (b) Label count distribution for USPTO. (c) Instance length distribution for USPTO.

contains all true labels including ancestors. For all results and analyses reported in this section, we modify the set of predicted labels to include relevant ancestors.

Prior work [14] has focused on evaluating per-instance (*micro*) scores. As the distribution of instances per label is highly skewed (see Fig. 3b), we additionally report *macro*-scores that average across scores obtained per label. We compute macro-F1 as the average over the macro-F1 scores per label. Unless otherwise stated, we consider a model to predict a label if the softmax score for the dimension "label applies" is at least 0.5. In addition, in line with previous work [14,38], we evaluate the predictions as a ranking task, which does not require defining a threshold. We compute the Area Under the Precision-Recall Curve (AUPRC) [7] as implemented in scikit-learn.[7] In the case of models outputting only leaf-level scores, we here use the maximum of the leaf-level scores for each intermediate-level label.

5.2 Implementation and Hyperparameter Settings

We implement our models in Python using TensorFlow 2.0[8] and Keras [6]. We use the HuggingFace Transformers library [39] for integrating SciBERT [3]. For efficiency reasons, we truncate the word-piece tokenized input sequences to a maximum length of 256. As illustrated in Fig. 3c, this covers the complete input text for almost all instances in USPTO (and also for WIPO-alpha, not shown).

We found the following hyperparameters to work best across our two benchmark datasets for our proposed TMM/THMM models. All dense layers have a hidden size of 256 and use ReLU activations. For training, we apply a learning rate of 10^{-5}, a dropout of 0.25 across layers, and a batch size of 64. In the case of the CNN model variant, we compute a single-vector document representation using a CNN whose architecture largely follows [22]. For each word-piece token, we compute an embedding by summing up the corresponding weights of the last four SciBERT layers. Then, we concatenate the embeddings of all word-piece tokens and apply convolution operations

[7] https://scikit-learn.org/stable/modules/generated/sklearn.metrics.average_precision_score.html
[8] https://www.tensorflow.org

with kernel sizes $\{2, 3, 4, 5\}$. In contrast to [22], we add an extra kernel of size 2 to capture bigrams and we use a filter size of 256 instead of 512. The training of a single model takes approximately 300 hours on a Nvidia Tesla V100 GPU with 80GB VRAM.

5.3 Baselines

We compare our models to a wide range of non-neural and neural models. First, the **TwistBytes** system [4] constitutes a competitive *non-neural* baseline. We run a recently updated version[9] leveraging a TF-IDF vector of uni-gram features. The system is implemented using scikit-learn[10] and learns one support vector classifier [31] per node. During prediction, the model only tests for presence of labels if the respective parent's score is positive. Finally, the set of predicted labels is filtered using a threshold of -0.25.

HARNN. In order to compare to a recent state-of-the-art *neural* model for hierarchical patent classification, we run the Hierarchical Attention-based Recurrent Neural Network [14] on our datasets. We keep hyperparameter settings as proposed, representing each document using a 100-dimensional Word2Vec [25] model trained on train and dev, using 256 and 512 as the hidden sizes in the BiLSTM and for each fully connected layer, respectively. Local and global information are combined with a regulation parameter α with a value of 0.5. For a fair comparison with the other models, we tune the prediction threshold for macro-performance, resulting in 0.15 for both datasets. **HARNN-orig** [14] uses a prediction threshold of 0.5.

flat-*. In addition, we provide results for simplified versions of our own model, predicting only labels for the leaf level and inferring ancestors during post-processing. First, **flat-CNN** corresponds to DeepPatent [22], which uses a CNN with kernels of sizes $\{3, 4, 5\}$ and 512 filters on top of SciBERT. The outputs of all CNN layers are flattened and concatenated, resulting in a 1,536-dimensional document embedding. Second, **flat-CLS** is based on PatentBERT [21], using SciBERT's 786-dimensional CLS embedding directly as document embedding. The feature vectors of **flat-CNN** and **flat-CLS** are subsequently fed into a multi-layer perceptron with two dense layers, applying sigmoid activation to each logit. For both models, dense layers have size 512, the learning rate is set to 10^{-5}, dropout rate is 0.25 and batch size is 64.

5.4 Experimental Results

We now analyze the performance of our models and compare them to prior work. We find similar tendencies on the two datasets and show that our models perform better especially at deeper levels of the hierarchy and for less frequent labels.

Classification Performance. Table 1 and Table 2 show results obtained for the USPTO and WIPO-alpha datasets, respectively. Based on these, we can draw the following conclusions. First, neural models generally perform better than the non-neural TwistBytes system, with SciBERT-based models outperforming HARNN. Our models achieve much

[9] https://dublin.zhaw.ch/~benf/HPC
[10] https://github.com/globality-corp/sklearn-hierarchical-classification

Table 1: Hierarchical classification results on **USPTO** test set.

	macro-avg.			**micro-avg.**			**AUPRC**
Model	**hP**	**hR**	**hF1**	**hP**	**hR**	**hF1**	
TwistBytes [4]	0.423	0.203	0.257	0.651	0.534	0.587	0.407
HARNN-orig [14]	0.355	0.126	0.170	**0.781**	0.481	0.595	0.661
HARNN [14]	0.292	0.281	0.267	0.519	0.679	0.588	0.661
flat-CNN [22]	**0.486**	0.272	0.330	0.718	0.552	0.624	0.645
TMM-CNN	0.412	0.360	0.366	0.639	**0.636**	0.637	0.667
THMM-CNN	0.412	0.364	0.369	0.649	0.634	0.641	0.669
flat-CLS [21]	0.481	0.256	0.316	0.740	0.546	0.628	0.644
TMM-CLS	0.485	0.313	0.362	0.709	0.611	**0.656**	**0.678**
THMM-CLS	0.426	**0.367**	**0.377**	0.666	0.633	0.649	0.670

higher recall while keeping precision high. When tuning HARNN for hF1 as in the original work, a high micro-hP can be achieved but at the cost of lower recall especially in the macro evaluation.[11] This implies that the original model focuses on the easy cases of highly frequent labels. Tuning HARNN for macro-scores changes the precision-recall tradeoff in the micro-setting and improves macro-F1, but still not approaching the performance of transformer-based models.

With the exception of macro-hP of flat-CNN on USPTO, the CLS-based models all outperform their CNN-based counterparts. However, the CLS-based models achieve the best results in terms of micro- and macro-F1 on both datasets. We conclude that there is no extra need for aggregating information across the sequence using a CNN layer. In most cases, adding hierarchical links between classification heads in TMM increases recall at the expense of precision. When comparing THMM-CLS with TMM-CLS on both datasets, the former does better in terms of macro-F1, while the latter has slightly higher micro-F1, i.e., adding the links helps especially for less frequent labels.

Finally, the flat strategy leads to good precision but is not competitive in terms of recall, illustrating that such models struggle with activating all relevant classifications to the required extent. The AUPRC scores also indicate that the TMM-CLS model performs best overall in terms of producing correct rankings of all labels for each patent, closely followed by THMM-CLS. Hence, our experiments confirm that when optimizing for a good trade-off between micro- and macro-average performance, hierarchical multi-label classification for patents is best approached using a fully hierarchical model.

Performance Across Levels. Fig. 4 shows an increase in macro-F1 for TMM and THMM compared to the baselines, resulting primarily from higher recall (not shown). Adding hierarchical links (THMM vs. TMM) results in better predictions mainly at level 3. Hence, the overall increase in F1 is a result of improved classification at the lower levels, and finer-grained labels benefit from passing on hierarchical information.

[11] We double-checked the surprisingly low macro-scores of HARNN-orig and decided to present results of HARNN tuned for macro-performance as well.

Table 2: Hierarchical classification results on **WIPO-alpha** test set.

	macro-avg.			micro-avg.			AUPRC
Model	**hP**	**hR**	**hF1**	**hP**	**hR**	**hF1**	
TwistBytes [4]	0.456	0.264	0.308	0.626	0.570	0.597	0.412
HARNN-orig [14]	0.089	0.021	0.027	**0.757**	0.248	0.373	0.505
HARNN [14]	0.206	0.269	0.206	0.373	0.652	0.474	0.505
flat-CNN [22]	0.466	0.348	0.382	0.707	0.578	0.636	0.641
TMM-CNN	0.408	0.400	0.389	0.636	0.684	0.659	0.681
THMM-CNN	0.377	0.413	0.380	0.620	0.686	0.651	0.674
flat-CLS [21]	**0.503**	0.328	0.377	0.737	0.598	0.660	0.674
TMM-CLS	0.462	0.376	0.399	0.682	0.679	**0.680**	**0.697**
THMM-CLS	0.409	**0.424**	**0.405**	0.651	**0.698**	0.674	0.690

Table 3: **Analysis of coverage for USPTO dataset.** *No Prediction*: number of test instances with no predicted labels at a given level. *False Positives (error analysis)*: average # hops between false positives and nearest true labels at third level.

	Avg. Labels Predicted			No Prediction			False Positives	
level	**1**	**2**	**3**	**1**	**2**	**3**	**# inst.**	**hops**
gold	1.56	1.89	2.32	0	0	0	0.0	0.0
TwistBytes [4]	1.65	1.51	1.56	147	891	1,575	3,788	4.17±1.77
HARNN-orig [14]	1.36	1.17	1.02	116	1,134	2,466	2,341	4.08±1.79
HARNN [14]	2.29	2.62	2.82	0	12	148	6,380	4.12±1.79
flat-CNN [22]	1.31	1.45	1.67	512	512	512	4,198	4.38±1.69
TMM-CNN	1.75	1.98	2.01	**1**	**42**	228	5,236	4.22±1.69
THMM-CNN	1.68	1.92	2.04	5	55	232	5,282	4.17±1.69
flat-CLS [21]	1.26	1.39	1.61	570	570	570	3,916	4.28±1.72
TMM-CLS	1.59	1.68	1.71	13	125	476	4,114	4.19±1.72
THMM-CLS	1.61	1.84	2.03	8	66	**204**	5,046	4.22±1.68

Coverage. The number of labels at the subclass (leaf) level varies strongly across instances from a single category to 20 or more, with a tendency of more recent patents having more labels. Hence, one difficulty of the task consists in outputting the right number of categories per instance [10]. Table 3 breaks down the average number of labels predicted by level of the hierarchy for USPTO (WIPO-alpha shows similar tendencies). At the top level of the hierarchy, all other models predict a roughly fitting number of labels. However, at levels 2 and 3, TwistBytes and the flat models predict markedly fewer labels. This effect is alleviated by the TMM and THMM models. While HARNN-orig strongly under-predicts the number of labels, our version of HARNN optimized for macro-F1 over-predicts, indicating that tuning the model either way is problematic. Next, we report the number of test instances for which a model did not make any prediction at a particular level ("No Prediction" in Table 3). This count is much lower for the TMM and THMM models, showing that the hierarchical models often can make predictions at intermediate levels even if the fine-grained class is unclear.

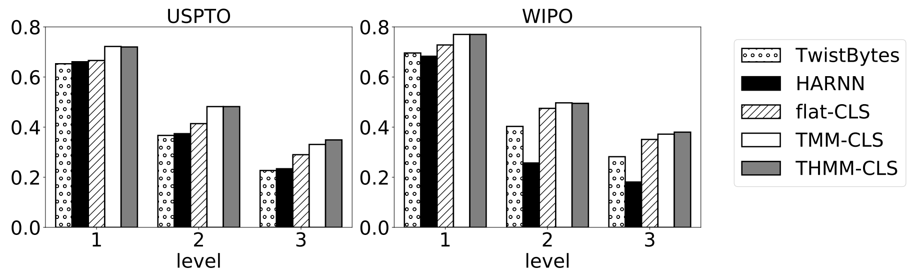


Fig. 4: Classification performance: **macro-avg. F1 by level of hierarchy**.

Error Analysis. Finally, we capture the models' mis-classification behavior by computing the number of hops in the label taxonomy to the nearest gold label on the same level. For example, if a model incorrectly predicts B41F and A43C with the true label being A43B, the wrong predictions are 6 hops and 2 hops away from the true label, respectively. Table 3 shows the average number of hops between false positives and gold labels on level 3. The column titled **# inst.** denotes the number of test instances having at least one false positive label. In general, the wrong predictions of all models seem to be similarly far from the nearest gold label, usually within the correct section of the taxonomy. Again, the flat approach more often activates completely wrong labels.

Summary. Our experiments on two patent datasets have shown that our models based on pre-trained transformers strongly outperform both neural and non-neural prior work in terms of micro- and macro-scores. Recall increased considerably while keeping precision high. The coverage of our models is much better than the one of prior work; wrongly activated predictions usually are within the correct section of the taxonomy.

6 Conclusion and Outlook

In this work, we have proposed a novel Transformer-based Multi-task Model (TMM) for hierarchical patent classification. The strength of our architecture stems from integrating the highly effective local-classifier-per-node idea from traditional hierarchical classification algorithms with a large-scale pre-trained neural transformer language model, which is made computationally feasible by our novel multi-task based architecture. We have shown that this model architecture strongly outperforms previous work on hierarchical text classification, with a higher coverage of instances and addressing the long tail of less frequent labels more successfully.

Future Work. Further improvements for patent classification can be expected from integrating additional textual information, e.g., the description or claims sections, for computing the document embedding. In this work, we have focused on patents. Yet, our model should be easily adaptable to other genres and domains, e.g., by substituting the pre-trained language model with in-domain data. Improving the confidence estimation for classification decisions further may lead to more precise label activation while keeping recall high. Finally, as our model has the very practical application of patent categorization, improving the model in an active learning set-up may be a very promising direction.

Acknowledgements We thank Mark Giereth and Jona Ruthardt for the fruitful discussions. We are grateful to Patrick Fievet for his support with the WIPO-alpha dataset. We also thank Alexander Müller for sharing his ideas on patent classification, and Lukas Lange, Trung-Kien Tran and the anonymous reviewers for their comments on this paper.

References

1. Abdelgawad, L., Kluegl, P., Genc, E., Falkner, S., Hutter, F.: Optimizing Neural Networks for Patent Classification. In: Brefeld, U., Fromont, E., Hotho, A., Knobbe, A., Maathuis, M., Robardet, C. (eds.) Machine Learning and Knowledge Discovery in Databases. pp. 688–703. Springer International Publishing (2019)
2. Aggarwal, C.C., Zhai, C.: Mining Text Data, chap. A Survey of Text Classification Algorithms, pp. 163–222. Springer (2012)
3. Beltagy, I., Lo, K., Cohan, A.: SciBERT: A pretrained language model for scientific text. 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). pp. 3615–3620. Association for Computational Linguistics, Hong Kong, China (Nov 2019)
4. Benites, F.: TwistBytes – Hierarchical Classification at GermEval 2019: walking the fine line (of recall and precision). In: Proceedings of the 15th Conference on Natural Language Processing (KONVENS). Erlangen, Germany (Oct 2019)
5. Benites, F., Malmasi, S., Zampieri, M.: Classifying Patent Applications with Ensemble Methods. In: Proceedings of the Australasian Language Technology Association Workshop 2018. pp. 89–92. Dunedin, New Zealand (2018)
6. Chollet, F., et al.: Keras (2015), https://github.com/fchollet/keras
7. Davis, J., Goadrich, M.: The Relationship between Precision-Recall and ROC Curves. In: Proceedings of the 23rd International Conference on Machine Learning. p. 233240. ICML '06, Association for Computing Machinery, New York, NY, USA (2006)
8. Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In: Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers). pp. 4171–4186. Association for Computational Linguistics, Minneapolis, Minnesota (2019)
9. D'hondt, E., Verberne, S., Oostdijk, N., Beney, J., Koster, C., Boves, L.: Dealing with temporal variation in patent categorization. Information Retrieval **17**, 520–544 (2014)
10. Fall, C.J., Törcsvári, A., Benzineb, K., Karetka, G.: Automated Categorization in the International Patent Classification. SIGIR Forum **37**(1), 1025 (2003)
11. Gomez, J.C., Moens, M.F.: A Survey of Automated Hierarchical Classification of Patents. In: Paltoglou, G., Loizides, F., Hansen, P. (eds.) Professional Search in the Modern World: COST Action IC1002 on Multilingual and Multifaceted Interactive Information Access, pp. 215–249. Springer International Publishing (2014)
12. Hepburn, J.: Universal Language Model Fine-tuning for Patent Classification. In: Proceedings of the Australasian Language Technology Association Workshop 2018. pp. 93–96. Dunedin, New Zealand (2018)

13. Howard, J., Ruder, S.: Universal Language Model Fine-tuning for Text Classification. In: Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). pp. 328–339. Association for Computational Linguistics, Melbourne, Australia (2018)
14. Huang, W., Chen, E., Liu, Q., Chen, Y., Huang, Z., Liu, Y., Zhao, Z., Zhang, D., Wang, S.: 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)
15. Jalan, R., Gupta, M., Varma, V.: Medical Forum Question Classification Using Deep Learning. In: Pasi, G., Piwowarski, B., Azzopardi, L., Hanbury, A. (eds.) Advances in Information Retrieval. pp. 45–58. Springer International Publishing (2018)
16. Kalchbrenner, N., Grefenstette, E., Blunsom, P.: A Convolutional Neural Network for Modelling Sentences. In: Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (ACL'14). pp. 655–665. Baltimore, MD, USA (2014)
17. Kang, D.M., Lee, C.C., Lee, S., Lee, W.: Patent Prior Art Search Using Deep Learning Language Model. In: Proceedings of the 24th Symposium on International Database Engineering & Applications. IDEAS '20, Association for Computing Machinery, New York, NY, USA (2020)
18. Kim, Y.: Convolutional Neural Networks for Sentence Classification. In: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP). pp. 1746–1751. Association for Computational Linguistics, Doha, Qatar (2014)
19. Kiritchenko, S., Matwin, S., Famili, A.F.: Functional annotation of genes using hierarchical text categorization. In: Proceedings of BioLINK SIG: Linking Literature, Information and Knowledge for Biology (2005)
20. Kowsari, K., Brown, D.E., Heidarysafa, M., Meimandi, K.J., Gerber, M.S., Barnes, L.E.: Hdltex: Hierarchical deep learning for text classification. In: 2017 16th IEEE International Conference on Machine Learning and Applications (ICMLA). pp. 364–371. IEEE (2017)
21. Lee, J.S., Hsiang, J.: PatentBERT: Patent classification with fine-tuning a pre-trained BERT model. World Patent Information **61**(101965) (2020)
22. Li, S., Hu, J., Cui, Y., Hu, J.: DeepPatent: patent classification with convolutional neural networks and word embedding. Scientometrics **117**(2), 721–744 (2018)
23. Lu, Z., Du, P., Nie, J.Y.: VGCN-BERT: Augmenting BERT with Graph Embedding for Text Classification. In: Jose, J.M., Yilmaz, E., Magalhães, J., Castells, P., Ferro, N., Silva, M.J., Martins, F. (eds.) Advances in Information Retrieval. pp. 369–382. Springer (2020)
24. Meng, Y., Shen, J., Zhang, C., Han, J.: Weakly-supervised hierarchical text classification. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 33, pp. 6826–6833 (2019)
25. Mikolov, T., Chen, K., Corrado, G., Dean, J.: Efficient Estimation of Word Representations in Vector Space. In: Bengio, Y., LeCun, Y. (eds.) 1st International Conference on Learning Representations, ICLR 2013, Scottsdale, Arizona, USA, May 2-4, 2013, Workshop Track Proceedings (2013)
26. Mikolov, T., Sutskever, I., Chen, K., Corrado, G.S., Dean, J.: Distributed representations of words and phrases and their compositionality. In: Advances in Neural Information Processing Systems. pp. 3111–3119 (2013)
27. Mollá, D., Seneviratne, D.: Overview of the 2018 ALTA shared task: Classifying patent applications. In: Proceedings of the Australasian Language Technology Association Workshop 2018. pp. 84–88. Dunedin, New Zealand (2018)
28. Nanba, H., Kamaya, H., Takezawa, T., Okumura, M., Shinmori, A., Tanigawa, H.: Automatic Translation of Scholarly Terms into Patent Terms. In: Proceedings of the 2nd International Workshop on Patent Information Retrieval. p. 2124. PaIR '09, Association for Computing Machinery, New York, NY, USA (2009)

29. Peng, H., Li, J., He, Y., Liu, Y., Bao, M., Wang, L., Song, Y., Yang, Q.: Large-scale hierarchical text classification with recursively regularized deep graph-cnn. In: Proceedings of the 2018 World Wide Web Conference. pp. 1063–1072 (2018)
30. Piroi, F., Hanbury, A.: Multilingual patent text retrieval evaluation: Clef–ip. In: Information Retrieval Evaluation in a Changing World, pp. 365–387. Springer (2019)
31. Platt, J.C.: Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. In: Advances in Large Margin Classifiers. pp. 61–74. MIT Press (1999)
32. Risch, J., Garda, S., Krestel, R.: Hierarchical document classification as a sequence generation task. In: Proceedings of the Joint Conference on Digital Libraries (JCDL). pp. 147–155 (2020)
33. Rogers, A., Kovaleva, O., Rumshisky, A.: A primer in bertology: What we know about how bert works. arXiv preprint arXiv:2002.12327 (2020)
34. Silla, C.N., Freitas, A.A.: A survey of hierarchical classification across different application domains. Data Mining and Knowledge Discovery **22**, 31–72 (2010)
35. Tang, P., Jiang, M., Xia, B.N., Pitera, J.W., Welser, J., Chawla, N.V.: Multi-label Patent Categorization with Non-local Attention-based Graph Convolutional Network. In: Proceedings of the Thirty-Fourth AAAI Conference on Artificial Intelligence (AAAI'20) (2020)
36. Tikk, D., Biro, G.: Experiment with a Hierarchical Text Categorization Method on the WIPO-alpha Patent Collection. In: Fourth International Symposium on Uncertainty Modeling and Analysis (ISUMA'03). pp. 104–109 (2003)
37. Wang, P., Fan, Y., Niu, S., Yang, Z., Zhang, Y., Guo, J.: Hierarchical Matching Network for Crime Classification. In: Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'19). pp. 325–334. ACM (2019)
38. Wehrmann, J., Cerri, R., Barros, R.: Hierarchical multi-label classification networks. In: Proceedings of the 35th International Conference on Machine Learning (ICML'18). pp. 5075–5084 (2018)
39. Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., Cistac, P., Rault, T., Louf, R., Funtowicz, M., Brew, J.: Huggingface's transformers: State-of-the-art natural language processing. ArXiv **abs/1910.03771** (2019)
40. Yao, L., Mao, C., Luo, Y.: Graph Convolutional Networks for Text Classification. In: Proceedings of the AAAI Conference on Artificial Intelligence (AAAI'19). pp. 7370–7377 (2019)