Hierarchical Document Classification as a Sequence Generation Task

Julian Risch
julian.risch@hpi.de
Hasso Plattner Institute,
University of Potsdam
Potsdam, Germany

Samuele Garda
garda@uni-potsdam.de
Hasso Plattner Institute,
University of Potsdam
Potsdam, Germany

Ralf Krestel
ralf.krestel@hpi.de
Hasso Plattner Institute,
University of Potsdam
Potsdam, Germany

ABSTRACT

Hierarchical classification schemes are an effective and natural way to organize large document collections. However, complex schemes make the manual classification time-consuming and require domain experts. Current machine learning approaches for hierarchical classification do not exploit all the information contained in the hierarchical schemes. During training, they do not make full use of the inherent parent-child relation of classes. For example, they neglect to tailor document representations, such as embeddings, to each individual hierarchy level.

Our model overcomes these problems by addressing hierarchical classification as a sequence generation task. To this end, our neural network transforms a sequence of input words into a sequence of labels, which represents a path through a tree-structured hierarchy scheme. The evaluation uses a patent corpus, which exhibits a complex class hierarchy scheme and high-quality annotations from domain experts and comprises millions of documents. We re-implemented five models from related work and show that our basic model achieves competitive results in comparison with the best approach. A variation of our model that uses the recent Transformer architecture outperforms the other approaches. The error analysis reveals that the encoder of our model has the strongest influence on its classification performance.

CCS CONCEPTS

• **Computing methodologies** → *Neural networks*; *Natural language processing*; • **Information systems** → **Document representation**; • **Social and professional topics** → Patents.

KEYWORDS

document classification; hierarchical classification; neural networks; deep learning; patent documents

ACM Reference Format:
Julian Risch, Samuele Garda, and Ralf Krestel. 2020. Hierarchical Document Classification as a Sequence Generation Task. In *ACM/IEEE Joint Conference on Digital Libraries in 2020 (JCDL '20), August 1–5, 2020, Xi'an, China.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/XXXXXX.XXXXXX

1 HIERARCHICAL CLASSIFICATION

Large document collections can typically be organized by a rich amount of topics and sub-topics. An effective and natural solution for organizing such collections is a hierarchical classification scheme. Hierarchical classes not only improve the retrieval of documents, but make it also easier to browse and explore the collection. However, classifying and indexing documents according to a complex hierarchy is time-consuming and involves experts with domain knowledge. Therefore, automatic approaches for hierarchical classification are crucial for reducing the burden of manual annotation and the latency of indexing.

In contrast to standard classification, where the classes are unrelated, hierarchical classification deals with schemes that are trees or directed acyclic graphs. The hierarchy imposes a parent-child relation among the classes. If an instance belongs to a specific class, it also belongs to all ancestors of that class.

One area that extensively uses hierarchical classification schemes is the patent domain. Patent applications need to be examined by patent officers. Among other criteria, these officers need to judge the novelty and the non-obviousness of an invention. This task requires domain-specific knowledge. To match patent applications with domain experts and in order to perform searches for related documents, patent offices draw on the standardized, hierarchical International Patent Classification (IPC) scheme. The IPC is used by over a hundred of patent-issuing bodies worldwide. Every year, it is revised and adapted to the upcoming of new fields of invention.

Figure 1 shows an exemplary classification. The scheme follows a tree-structure, which means each inner node in the hierarchy has exactly one parent. The label *F02D 41/02* comprises the section *F,* the class *F02,* the subclass *F02D,* the main group *F02D 41* and the subgroup *F02D 41/02.* In its version of 2018, the IPC contains in total 8 sections, 131 classes, 642 subclasses, 7,537 groups and 69,487 subgroups. At each level of the hierarchy, the labels also have a description. For instance, *F02*, corresponds to "combustion engines" and its subclass *F02D* to "controlling combustion engines".

The upcoming of deep learning and its success at natural language processing tasks inspired many new approaches for document classification in general and patent classification in particular [1, 17, 24]. However, no previous work uses sequence-to-sequence neural network models to address hierarchical patent classification. These models are typically used for machine translation and speech to text applications. They comprise an encoder and a decoder. The encoder processes a sequence of inputs, such as written or spoken words. It compresses the information into a (context) vector representation. The decoder is initialized with this

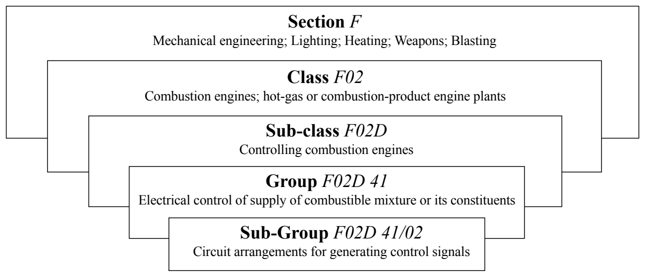
Section *F*
Mechanical engineering; Lighting; Heating; Weapons; Blasting
Class *F02*
Combustion engines; hot-gas or combustion-product engine plants
Sub-class *F02D*
Controlling combustion engines
Group *F02D 41*
Electrical control of supply of combustible mixture or its constituents
Sub-Group *F02D 41/02*
Circuit arrangements for generating control signals

Figure 1: The International Patent Classification (IPC) scheme is a complex, hierarchical scheme for patent documents.

vector representation and emits a sequence of outputs, such as the translated words.

In this paper, we show that classifying a document according to a hierarchical scheme corresponds to generating a sequence of class labels. Therefore, we propose a sequence-to-sequence neural network model that takes a document text as input and outputs a sequence of class labels — starting with the highest (broadest) level in the hierarchical classification scheme and ending with the most fine-grained level. Further, the proposed model internally uses label embeddings. For example, when the generated sequence begins with the section *F* and the class *F02,* the embeddings of these two labels are taken into account for generating the next, more fine-grained label. Label embeddings help to learn internal representations of classes that only have a few training samples. They are also used in the field of few-shot learning.

Contributions. In summary, we make the following contributions: First, we describe how to address hierarchical classification as a sequence generation task and show the benefits of this method. Second, we design and implement a transformer-based neural network architecture that follows the sequence-to-sequence paradigm. Finally, we evaluate this model in comparison to five re-implemented approaches from related work on a patent classification task and analyze its errors. To the best of our knowledge, no previous work explores sequence-to-sequence models for the hierarchical classification of patent documents.

Outline. Section 2 summarizes related work in the field of hierarchical classification and gives an overview of approaches for patent classification in particular. In Section 3, we introduce our approach and explain how to address hierarchical classification as a sequence generation task. Further, we describe variations of our model's network architecture and its training process, such as scheduled sampling and beam search. We evaluate our approach with experiments on a dataset of patent documents in Section 4 and conduct an error analysis in Section 5, before we conclude in Section 6.

2 RELATED WORK

Hierarchical classification approaches span three categories: flat, local, and global. Flat approaches *flatten* the class hierarchy and completely ignore parent-child relations. They assume a standard multi-class classification problem and work only on the level of leaf nodes. Therefore, they are inefficient for hierarchies with many levels and classes.

Local approaches follow the top-down paradigm. They classify the top level first and all other levels subsequently. To this end, one independent classifier is trained for each node in the hierarchy (except for leaf nodes). The classifiers share no information among them. As a consequence, classifiers at the bottom of the hierarchy suffer from sparse training data. Further, errors propagate from upper-level classifiers to subsequent ones.

Global approaches use only one classifier. Several studies on global approaches propose to directly integrate the hierarchical classification scheme in the learning process, both in the standard machine learning [9, 31] and in the deep learning [21, 32] framework. They modify the model architecture or they design loss functions tailored to the task of hierarchical classification. An advantage over flat and local approaches is that they avoid irrecoverable errors on upper classification levels and that they leverage the parent-child relations during training.

However, global approaches still do not fully exploit hierarchical classification schemes. The main issue is that they cannot refine the document representation (feature set) while they proceed in the classification. Throughout the classification process and at each hierarchy level, the same, unaltered document representation is used.

The reason for this is the way the output space is defined. In general, the global approach defines an output space or a loss function that models the label dependencies statically. Although penalized when producing a wrong chain of predictions (from the root to a leaf node), the model never has the chance to improve the document representation for the specific node in the hierarchy. Therefore, we propose a model that is able to modify the document representation at each step and thereby consider the relevant aspects of a document for the respective hierarchy level. This ability significantly improves the performance on hierarchical classification tasks.

Precursors of Sequence-to-Sequence Models. Caled et al. address a hierarchical classification task with a neural network model [7]. Their model outputs three labels, which correspond to different hierarchy levels. Although each output serves as input to the other outputs for more fine-grained hierarchy levels, the model does not completely follow a sequence-to-sequence architecture. One single internal representation is used for the three outputs. As a consequence, the representation cannot be tailored to the different hierarchy levels. We address this drawback with our approach and use one vector representation for each level. Wehrmann et al. work on hierarchical multi-label classification [28]. Their neural network combines a global and a local approach by using multiple loss functions. Similar to the approach by Caled et al., their model has no sequence-to-sequence architecture. The model cannot properly learn and exploit the interdependencies of class labels in the hierarchy.

Sequence-to-Sequence Models. Yang et al. propose a sequence-to-sequence model for multi-label classification [33]. However, the labels that they consider have no hierarchy. While their model outputs an ordered sequence, the ground truth labels of each training sample are given as an unordered set. This discrepancy limits the applicability of sequence-to-sequence models to the multi-label classification task, and requires a different method of calculating the training loss [26]. Li et al. propose a sequence-to-sequence model for hierarchical classification [16]. When their model generates an output sequence, they do not enforce the class hierarchy. Instead, they accept if their model makes up a new class that extends the pre-defined hierarchy. While this feature can be helpful for revising hierarchies in unconstrained scenarios, it is inappropriate for the patent domain. Therefore, we ensure that our model generates only those label sequences that agree with the hierarchy.

Label Embedding. None of the above mentioned approaches for hierarchical classification uses label embeddings. These embeddings are a part of our approach and therefore, we briefly describe related work on label embeddings in the following. Related work on document classification extends the ideas of word embeddings [6, 19, 22] and document embeddings [15, 30] to also include label embeddings. As for words and documents, dense vectors can represent labels in a high-dimensional space to capture their semantics. This method is particularly successful at the task of few-shot and zero-shot learning [18, 34], where the relations between labels captured in the embedding space improve the prediction for classes that rarely or never occur in the training data.

Several approaches tailor label embeddings to multi-class and multi-label classification tasks. They adapt the loss functions in the training process of neural networks [20], use multiple training tasks [8], and jointly learn word, sentence, document, and label embeddings [3, 27]. All these approaches neglect class hierarchies. They assume a flat output space with independent classes.

Patent Classification. Fall et al. [10] introduce standard evaluation tasks and metrics for automated patent classification. We explain and use these metrics in the evaluation section. An overview of approaches for patent classification can be found in a survey by Gomez et al. [11]. The most relevant approaches are naive Bayes, support vector machines, and k-nearest neighbors. They are applied to a series of CLEF-IP tasks [23].

More recently, deep learning is used for patent classification. Grawe et al. use long-short term memory units (LSTMs) but relax the task by limiting the number of classes to 50 [12]. Another approach with recurrent neural networks uses gated recurrent units (GRUs) and patent-specific word embeddings [24]. Li et al. introduce a benchmark dataset comprising two million patents from the United States Patent and Trademark Office, called USPTO-2M [17]. Their approach is based on a convolutional neural network (CNN) by Kim [13]. To the best of our knowledge, there are no publications that focus on issues of hierarchical classification in the patent domain.

3 READ, ATTEND AND LABEL

This section describes how we address hierarchical classification as a sequence generation task. Further, we introduce a neural network architecture that implements this idea following a discussion on its characteristics. Referring to the "show, attend and tell" approach [30], we call our approach "read, attend and label" (RAL). Our source code is publicly available online.[1]

[1]https://hpi.de/naumann/projects/repeatability/text-mining.html

3.1 Hierarchical Labels as Sequences

In a hierarchical classification scheme, a leaf node at the bottom of the hierarchy describes the lowest, most fine-granular label. For example, label *F02D 41/02* is a leaf node in the introductory example from the patent domain. This *explicit label assignment* also entails a group of *implicit label assignments*, which is the chain of ancestors in the tree-structured scheme. To this end, the *explicit label assignment* defines parent-child relations that represent *implicit label assignments*, symbolized by $\prec$ (read "implies"):

$$F02D\ 41/02 \prec F02D\ 41 \prec F02D \prec F02 \prec F$$

Each level in the hierarchy represents a degree of specialization for a given label, which shares some properties with its ancestors. The group of *implicit label assignments* essentially is the path from the root to a specific leaf node in the tree-structured hierarchical scheme. We understand this path as a *sequence* of assignments at different levels. The task of finding the correct leaf node for a given document can be addressed as the task of generating the sequence of inner nodes that form a path from the root to the correct leaf node.

To this end, given a document, we define a model whose objective function is to minimize the reconstruction error of the sequence of label assignments. Instead of using individual models per hierarchy level, we use one holistic model. It first assigns a label at the top (broadest) level and then, step by step, at each subsequent level down to the leaf nodes. Intuitively speaking, the model emulates the process that patent officers conduct. Instead of immediately coming up with the correct leaf node *F02D 41/02*, the classification process is iterative and first assigns the section *F*, second the class *F02*, and so on. Our model undergoes the same iterative process when generating a sequence of labels and therefore can learn:

- the sequence structure, e.g., each sequence begins with one out of eight letters that represent the section;
- the label relations, e.g., the explicit label *F02D 41/02* implies the implicit label *F* and they share some characteristics;
- the mutual exclusiveness of certain labels, e.g., if the generated sequence starts with section *F* it cannot continue with class *A01* or with section *B*.

3.2 Sequence-to-Sequence Model

We propose a special kind of sequence-to-sequence neural network model inspired by the work of Xu et al. on the task of image captioning [30]. For our model, the sequence of words in a patent's text serves as the input and the generated output is a sequence of labels, which describes a path from the root to a leaf node in the tree-structured hierarchical classification scheme. Figure 2 shows our neural network architecture. Both the encoder and the decoder differ from the ones by Xu et al. In addition to convolutional layers, fastText word embeddings encode the sequence of input words [6]. The output sequence does not consist of words that represent an image caption, but of labels.

The inputs to the decoder are label embeddings, which are embedded in the same space as the input words in the encoder. Therefore, the labels can leverage the semantics of the words in their text descriptions. For instance, the descriptions "combustion engines" of

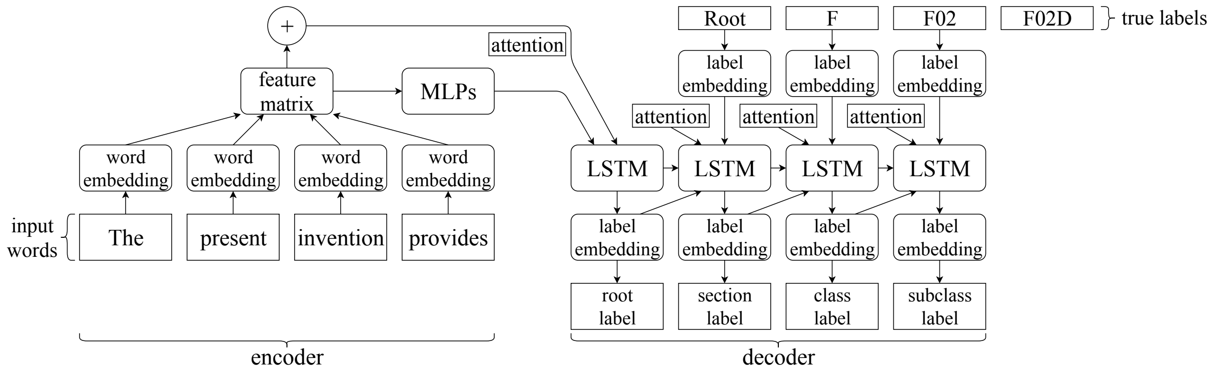

Figure 2: Our sequence-to-sequence architecture for hierarchical classification consists of an encoder and a decoder part.

class *F02* and "machines or engines for liquids" of class *F03* are semantically similar because the embeddings of the respective words are similar. To also incorporate information from the hierarchical scheme, a label's embedding considers its own description and the descriptions of its ancestor nodes. This is achieved by taking the average of all the word embeddings of its description and the label embeddings of its ancestor nodes. For example, the label embedding of class *F02* is the average of the word embeddings of the description of *F02* and the label embeddings of section *F*.

The attention mechanism in the neural network controls how relevant each word in a patent document is for the label prediction at the different levels of the hierarchy. When a sequence of section, class, and subclass labels is generated, each label is generated based on a differently weighted sum of the input words.

We now describe the details of the network architecture in a more formal way. The encoder maps the input words to their pretrained word embeddings. *F* is the *feature map*, which is a matrix that holds the word embeddings. After encoding the sequence of input words in this way, two separate multilayer perceptrons are used to initialize the memory state c_o and the hidden state h_o of the LSTM cell in the decoder. The equations 1 and 2 show this initialization:

$$c_o = \tanh(W_c \cdot \frac{1}{N} \sum_{i}^{N} F_i + b_c) \tag{1}$$

$$h_o = \tanh(W_h \cdot \frac{1}{N} \sum_{i}^{N} F_i + b_h) \tag{2}$$

N is the number of input words, b_c and b_h are trainable bias terms, and W_c and W_h are trainable weight matrices of the multilayer perceptrons.

The attention mechanism allows for each input word being of different relevance for the different hierarchy levels of prediction (e.g., section, class or subclass). The context vector $S_{(t)}$ is produced with the previous hidden state and the word embedding matrix F (internal memory). This vector can be interpreted as a summarization of the input document with the relevant words emphasized for the given level of prediction.

More formally, the attention mechanism can be considered as the model's *internal memory*. The memory is defined as a matrix, i.e. the representation obtained by the encoder for each element in the input sequence. We use an attention mechanism by Bahdanau et al., which is called *soft attention* [2]. At each time step t in the decoding process, in order to generate the next output label, given the decoder hidden state $h_{dec}^{(t)}$ and the internal memory $h_{enc}^{(1,\cdots,N)}$, a multilayer perceptron is used to calculate the attention weights:[2]

$$\text{score}(h_d^{(t)}, h_e^{(1,\cdots,N)}) = \tanh(W_d \cdot h_d^{(t)} + W_e \cdot h_e^{(1,\cdots,N)}) \tag{3}$$

where d stands for decoder, e for encoder, W_d and W_e are the parameters of the feed-forward layer, and $tanh$ is the activation function. Given this scoring function, it possible to calculate the alignment scores, i.e., to what extent the output label to be generated and the words in the input sequence are *aligned*. To this end, the softmax function is applied to the result of the alignment:

$$\alpha_t = \text{softmax}(\text{score}(h_d^{(t-1)}, h_e^{(1,\cdots,N)})) \tag{4}$$

The resulting vector α_t has the same length as the input sequence. Finally, in order to obtain the context vector, the rows in the model's internal memory (the vectors corresponding to the input words) are weighted by the values of α_t and then summed:

$$c_t = \sum_{i=1}^{N} \alpha_{t,i} \cdot h_e^i \tag{5}$$

The target sequence is represented by the label embeddings $L_{(t)}$ from the hierarchical classification scheme. $S_{(t)}$ and $L_{(t)}$ are concatenated and then fed to the LSTM cell. The hidden state of the LSTM, $H_{(t)}$, can now be interpreted as the document embeddings for the given input document. Finally, $H_{(t)}$ is fed to a multilayer perceptron in order to generate the next output label.

3.3 Scheduled Sampling

When the model generates a label embedding during training, it is provided with the true label embedding of the previous step instead of its own prediction of the previous step. This training technique is known as *teacher forcing*. As a consequence during training, the model is never exposed to its own mistakes. It suffers from an *exposure bias*. At test time however, the true labels are not available

[2]In machine translation, this multilayer perceptron is also known as the alignment model.

and the model is not prepared for that difficulty. In order to mitigate this issue, we use the *scheduled sampling* mechanism as proposed by Bengio et al. [4]. At the very beginning of the training process, the model is provided with the true label embeddings *(teacher forcing)*. As the training proceeds, the true label embeddings are more and more often not provided and instead only the model's own predicted (sampled) label embeddings are used. The probability of using the true labels follows an inverse sigmoid decay. More precisely, it is the inverse of the sigmoid decay at the i-th training step, where k is a parameter typically chosen in the range of the number of training steps:

$$\frac{k}{k + \exp(\frac{i}{k})}$$

3.4 Beam Search

At prediction time, the decoder outputs a sequence of vectors, each representing a probability distribution over all possible labels (the vocabulary). This is achieved by a dense layer with softmax activation that takes as input a label embedding. Choosing the most likely output label at each step of the output sequence generation is a greedy approach. As an improvement to this naive algorithm, we use beam search and follow a fixed number (the beam width) of labels with highest probability. We apply a beam search variant that is specifically designed for the sequence-to-sequence architecture [29]. It introduces a *coverage penalty*, which penalizes beams that do not cover the full source sequence. The beam width is set to eight so that all eight possible section labels are taken into account for the first decoding step (the first level of the hierarchy). The beam search approach allows the model to generate a number of most likely label sequences. In contrast, the greedy approach can only generate one sequence, selecting always the most likely label.

More formally, the Beam Search variant is defined as:

$$\mathrm{s}(X, Y) = \log(P(Y|X)) + \mathrm{cp}(X; Y) \tag{6}$$

where $P(Y|X)$ is the probability of the target sequence given the source sequence and *cp* stands for coverage penalty. The penalty is formulated as follows:

$$\mathrm{cp}(X; Y) = \beta * \sum_{i=1}^{|X|} \log(\min(\sum_{i=1}^{|Y|} p_{i,j}, 1)) \tag{7}$$

where $p_{i,j}$ is the attention probability of the j-th target word y_i on the i-th source word x_i. This penalty encourages generation of an output sequence that is most likely to cover all the words in the source sentence.

3.5 Benefits

The proposed model improves on the standard hierarchical classification models in several ways. First, the model directly encodes the parent-child relation and the path through the tree-structured hierarchy into the labels. This improvement is achieved by defining the label embeddings as a composition of their own text description and the text description of their ancestors. Second, it improves the representation of sparsely populated leaf nodes. If the label *F02D* has very few training samples, the model still learns characteristics of *F02,* because it shares the training data of samples of *F02B, F02C,* etc. Further it has a representation for *F02D* even without any training samples but solely based on the label embedding of *F02D*. Third, and most importantly, the model is able to refine the document representation and the label embedding at each level of the hierarchy.

Table 1: Training, validation and test set size of USPTO-2M.

	#Documents	#Terms	#Tokens	#Labels
Training	1,900,347	99,609	1,602,406	632
Validation	49,900	31,117	302,892	614
Test	49,900	19,220	303,088	606
	2,000,147	104,267	2,602,406	632

3.6 Transformer

Now that we introduced our sequence-to-sequence architecture and its training procedure, we propose a variation. To this end, we replace the document representation part with the state-of-the-art Transformer model [25]. The Transformer model uses a self-attention mechanism in six layers of encoders and decoders. Further, it adds positional encoding and as a consequence the order of the input words becomes relevant.

This variation makes the document representation more complex and entails minor changes to the training procedure. Scheduled sampling does not apply to the Transformer model. When the Transformer model generates a label embedding during training, it is provided its own predictions from *all* previous steps and not only its own prediction of the previous step. However, the beam search works without any changes. All benefits of the standard RAL model also apply to the presented variation of RAL, which we refer to as the *Transformer model* throughout the rest of this paper.

4 EVALUATION

We evaluate our model on a hierarchical classification task and compare its performance to four approaches from related work and a naive Bayes baseline. The dataset, evaluation metrics, and experiments are described in the following.

4.1 Dataset

The United States Patent and Trademark Office (USPTO) provides a collection of more than 9 million patents on their website. Li et al. extracted a subset of 2,000,147 patents, which they call UPSTO-2M [17]. They pruned the label to the subclass level, resulting in 632 different labels. The dataset only includes the title and the abstract because these two fields are generally considered the most informative parts of patents for the classification task [5]. To model a realistic evaluation scenario, where the fields of inventions evolve over time, we apply a time-wise split into training, validation, and test set. To this end, the training dataset comprises patents published between 2006 and early 2014, the validation set comprises patents from late 2014, and the test comprises patents from 2015. Table 1 lists the sizes of the different subsets. Figure 3 shows the class distribution on the highest level in the hierarchical classification scheme. The distribution differs only slightly between validation and test set.

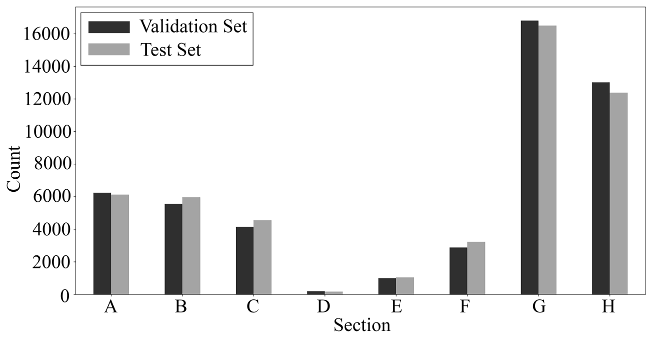

Figure 3: The class distribution on the *section* level of the class hierarchy is highly imbalanced. However, it differs only slightly between validation and test set.

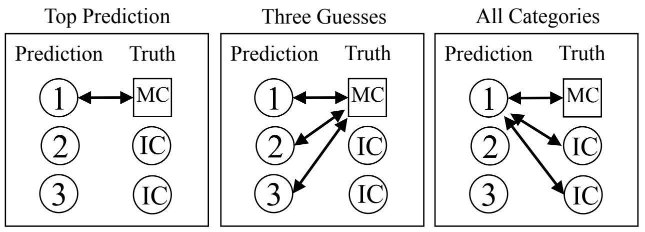

Figure 4: Three conditions defining micro-average precision for the task of patent classification. The predicted ranked classes are compared to the ground truth main category (MC) and incidental categories (IC). In this paper, *categories* are labels on the subclass level of the IPC scheme. The figure was adapted from Fall et al. [10].

4.2 Metrics

Patent classification is typically evaluated with *micro-average precision* as proposed by Fall et al. [10]. This measure is tailored to three practical application scenarios of patent classification. The first one is called Top Prediction (TP) and it compares the top prediction (main category) to the ground truth. This formulation is identical to accuracy. Note that the class distribution in our particular dataset and in patent datasets in general is imbalanced. Models that always predict the majority class can achieve a high accuracy. While this metric can be misleading, we include it to be comparable to related work and rely on other metrics to draw conclusions.

The second metric is called Three Guesses (TG) and is a relaxed version of TP. The three top-ranked predictions are compared to the top true label. Finally, the last condition is All Categories (AG), which considers instead the top three true labels assigned to a patent (main category and incidental categories) as the ground truth. The prediction is considered correct if it is in the set of the three true labels. Figure 4 visualizes the three scenarios.

Hierarchical evaluation measures not only evaluate the predicted leaf node, but also the prediction on upper levels in the hierarchical classification scheme. Even if the leaf node is predicted wrong, upper levels might be correct, which is better than a completely wrong prediction (a completely wrong path through the tree-structured hierarchy).

Table 2: Scheduled sampling improves the model's classification results according to five out of six evaluation metrics on the validation set.

Sampling	Acc	TG	AG	LCA-P	LCA-R	LCA-F1
standard	53.5	**75.8**	64.8	70.8	63.2	66.0
scheduled	**54.5**	75.1	**65.7**	**71.2**	**63.6**	**66.4**

Table 3: A GLU encoder with kernel (k) size of 3, filter (f) size of 256 and depth (d) of 2 outperforms all other tested parameter settings and also the model without GLU encoder (base) on the validation set.

	Acc	TG	AG	LCA-P	LCA-R	LCA-F1
base	52.0	74.5	63.1	70.0	62.3	65.1
f128_k3	52.2	74.5	63.2	70.0	62.4	65.3
f256_k3	52.3	74.3	63.2	69.9	62.4	65.2
f512_k3	52.1	74.0	63.2	70.0	62.4	65.2
f256_k3_d2	**52.9**	**75.0**	**64.2**	**70.3**	**62.8**	**65.6**
f256_k3_d3	52.6	74.8	63.8	70.2	62.7	65.5

A common measure for hierarchical document classification is the Lowest Common Ancestor F-measure (LCA-F1) introduced by Kosmopoulos et al. [14]. LCA-F1 overcomes two issues that are present in other hierarchical measures. Most other hierarchical measures consider the intersection of the sets of all ancestors of the predicted class and of the true class. However, taking into account all ancestors has the undesirable effect of over-penalizing errors that happen to nodes with many ancestors. Moreover, these measures suffer from the so called "long distance problem", where the predicted class is on the same level as the true class, but in a wrong branch of the tree. The LCA measures (LCA-Precision, LCA-Recall, LCA-F1) consider the lowest common ancestor for the evaluation. Thereby, it takes into account a very limited set of ancestors and correctly penalizes predictions according to the hierarchical classification scheme. We focus on LCA-F1 for our evaluation but also list accuracy, micro-average precision for *three guesses* and *all categories*, and LCA-Precision and LCA-Recall for the sake of completeness.

4.3 Hyperparameter Optimization

We tune the cell size of the LSTM units (128, 256, 512), the size of the attention layer (128, 256, 512), the recurrent dropout rate (0.1, 0.2, 0.3) on the validation set. Cell size and attention layer size are both set to 256 and dropout rate to 0.2. The training uses the Adam optimizer and the number of epochs is set to five after experiments on the validation set. Given five training epochs, a batch size of 32, and 1.9 million training samples, there are about 300,000 training steps. For the scheduled sampling, the parameter k of the inverse sigmoid decay is therefore set to $1.3 \cdot 10^4$. Table 2 shows that the scheduled sampling improves the predictions on the validation set. Similarly, Table 3 and Table 4 explain the choices of the other hyperparameters.

Table 4: A recurrent dropout rate of 0.2 performs best on the validation set, while rates of 0.1 or 0.3 yield similar results.

	Acc	TG	AG	LCA-P	LCA-R	LCA-F1
rd0.0	52.9	75.0	64.2	70.3	62.8	65.6
rd0.1	53.4	75.5	64.7	70.7	63.1	**66.0**
rd0.2	**53.5**	75.8	**64.8**	**70.8**	**63.2**	**66.0**
rd0.3	53.3	**75.9**	**64.8**	70.7	63.1	**66.0**

Table 5: Patent classification results on the test set sorted by lowest common ancestor F1 (LCA-F1). We also list accuracy (Acc), micro-average precision for *three guesses* (TG) and *all categories* (AG), and the hierarchical metrics LCA-Precision and -Recall.

					LCA-	
Model	Acc	TG	AG	P	R	F1
Naive Bayes	40.1	56.2	53.3	62.2	49.7	54.1
Tree-CNN [32]	40.0	63.2	60.2	66.8	53.2	57.9
CNN [17]	45.5	67.0	63.4	67.2	55.0	59.5
LEAM [27]	51.9	75.5	70.0	70.7	57.1	61.9
RAL	53.2	74.9	70.4	70.5	57.9	62.5
GRU [24]	54.0	77.3	72.7	**72.2**	58.4	63.3
Transformer	**56.7**	**78.9**	**74.5**	72.1	**59.5**	**64.2**

4.4 Results on the Test Set

On the test set, we compare RAL and the Transformer model to a naive Bayes baseline that uses bag-of-words features and four other approaches from related work:

- Tree-CNN by Yan, which is a convolutional neural network tailored to hierarchical classification and the biomedical domain [32]
- CNN by Li et al., which is a convolutional neural network for patent classification [17]
- LEAM by Wang et al., which is a label embedding approach for text classification in general [27]
- GRU by Risch and Krestel, which is a recurrent neural network for patent classification [24]

The models use 100-dimensional fastText word embeddings that were pre-trained on a patent corpus [24].

Table 5 lists the experiment results on the test set. Our approach, RAL, outperforms the baseline and three of the four approaches from related work. It achieves competitive results with the fourth approach, which is the GRU. The Transformer model outperforms all approaches. Its strong performance indicates that a sophisticated document representation is of advantage.

5 ERROR ANALYSIS FOR THE RAL MODEL

The weights of the attention layer give an insight into the errors of our model. A heat map visualizes the weights with the patent abstract's word on the horizontal axis (excluding stopwords) and the generated labels on the vertical axis. Each cell in the grid represents the amount of attention a word receives when the corresponding

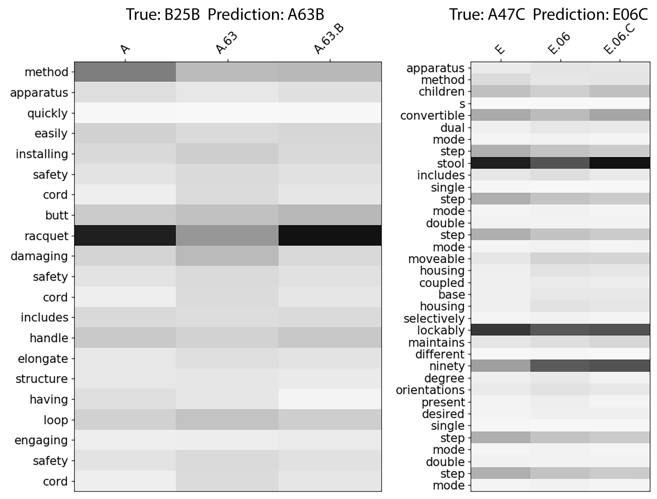

Figure 5: Attention weights for patent abstract with true label *B25B* and prediction label *A63B* (left) and with true label *A47C* and prediction label *E06C* (right).

label is predicted. The misclassified examples confirm what the hierarchical measures in the evaluation suggested. If the model makes an error, it misses the right label completely, e.g., it predicts *B01D* instead of *E04C*.

The left-hand side of Figure 5 represents one example. The model assigns label *A63B* which is composed by:

- section *A:* Human Necessities
- class *A63:* Sport, Games, Amusement
- subclass *A63B:* Apparatus for physical training, gymnastics, swimming, climbing, or fencing; ball games; training equipment

The model focuses its attention on the word *racquet*, which is indeed a "hand tool" used to play tennis. The model's decision to assign section *A* was essentially based on this single information and the word *method.* While for the class prediction *(A63)* the word *racquet* was not as relevant, it was again crucial for the final prediction of the subclass *(A63B).*

Based on this information the model prediction seems reasonable. Nonetheless, the ground truth label for the abstract was instead *B25B*, which is composed of:

- section *B:* Performing operations, Transporting
- class *B25:* Hand tools; Portable power-driven tools; Handles for hand implements; Workshop equipment; Manipulators
- subclass *B25B:* Tools or bench devices not otherwise provided for, for fastening, connecting, disengaging, or holding

This error and all the one akin underline an essential flaw in the model. The model is prone to misclassification at the first level of the hierarchy. Given the shortness of the sequence, the model is bound to its first decision. Error irrecoverably propagate to the following hierarchy levels and predictions.

The main reason for this behavior is imputable to the document representation obtained via the attention mechanism. This component of the model is essentially a feature extraction method. The document representation is simply a weighted sum of the document's words. Nonetheless, for this task, predicting the label simply based on the presence of specific words and their composition proves to be ineffective.

More precisely, the ability of correctly encoding the syntactic structure of the abstracts is crucial for generating the correct prediction. The example presented above shows that the model fails to encode that the main subject of the document is a "method/apparatus" that allows to install a "safety cord" on the "butt" of a "racquet", hence *B25B* being the correct class. Further evidence for this hypothesis is the performance of the CNN by Li et al. [17], which is significantly lower than the GRU network [24] in our evaluation. This is because the convolutional layers essentially perform a feature extraction operation similar to the one of the attention mechanism, e.g., activating the most relevant words but failing to capture the syntax of the document. With its more sophisticated document representations, the Transformer model outperforms all other approaches as listed in Table 5.

Additionally, the right-hand side of Figure 5 presents a similar — but more subtle — example of the errors caused by the attention mechanism. The true label for the document is *A47C,* which is composed by:

- section *A:* Human Necessities
- class *A47:* Furniture; Domestic articles or appliances; Coffee mills; Spice mills; Suction cleaners in general
- subclass *A47C:* Chairs, Sofas, Beds

The document describes a "stool" (chair) designed for children, which presents the feature of being convertible, e.g., it allows to increase the height of the "stool" ("dual mode") as the children grow. Although the model correctly focuses its attention mostly on the word "stool" for the section level, its prediction is *E06C,* which is composed by:

- section *E:* Fixed constructions
- class *E06:* Doors, Windows, Shutters, or Roller blinds in general; Ladders
- subclass *E06C:* Ladders

The model is confused by words that correlate with innovations in the ladders field, such as "lockably" and "ninety". The word "steps" appears five times in the document. Although these words do not receive much attention, in sum they affect the final document representation, being more related to "ladders" than to "chairs".

6 CONCLUSIONS AND FUTURE WORK

We are the first to tackle the task of hierarchical classification as a sequence generation problem. To this end, we implemented an attention-based neural network model that follows the sequence-to-sequence paradigm. Given a document's text representation, the model generates a sequence of class labels, which can be seen as a path through a tree-structured hierarchical classification scheme. This model leverages parent-child relations for its training and is the first to adapt the document representation for each level of the hierarchy individually. The underlying label embeddings help the model to cope with sparse training data.

We compare our model to five re-implemented approaches from related work on a dataset of patents, which exhibits a hierarchical classification scheme. The evaluation shows that our model RAL outperforms three of the four other approaches and that it achieves competitive results with the fourth model. Further, we find that a variation of our approach, which uses a different representation based on the Transformer model, outperforms all other approaches. The error analysis reveals that recovering from early errors during the sequence generation is difficult even for the best models. The most promising direction for future work therefore is to combine the sequence-to-sequence architecture with more powerful encoders and more complex document representations. Another direction is to use the label embeddings and the attention weights for interpretability of the otherwise black-box neural networks.

REFERENCES

[1] Louay Abdelgawad, Peter Kluegl, Erdan Genc, Stefan Falkner, and Frank Hutter. 2019. Optimizing Neural Networks for Patent Classification. In *Proceedings of the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML-PKDD).* 16.

[2] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *Proceedings of the International Conference on Learning Representations (ICLR).* 1–15.

[3] Simon Baker, Douwe Kiela, and Anna Korhonen. 2016. Robust text classification for sparsely labelled data using multi-level embeddings. In *Proceedings of the Conference on Computational Linguistics (COLING).* 2333–2343.

[4] Samy Bengio, Oriol Vinyals, Navdeep Jaitly, and Noam Shazeer. 2015. Scheduled sampling for sequence prediction with recurrent neural networks. In *Advances in Neural Information Processing Systems (NeurIPS).* 1171–1179.

[5] Karim Benzineb and Jacques Guyot. 2011. Automated patent classification. In *Current Challenges in Patent Information Retrieval.* Springer, 239–261.

[6] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics (TACL)* 5 (2017), 135–146.

[7] Danielle Caled, Miguel Won, Bruno Martins, and Mário J. Silva. 2019. A Hierarchical Label Network for Multi-label EuroVoc Classification of Legislative Contents. In *International Conference on Theory and Practice of Digital Libraries (TPDL)*, Antoine Doucet, Antoine Isaac, Koraljka Golub, Trond Aalberg, and Adam Jatowt (Eds.). Springer, 238–252.

[8] Sheng Chen, Akshay Soni, Aasish Pappu, and Yashar Mehdad. 2017. Doctag2vec: An embedding based multi-label learning approach for document tagging. *arXiv preprint arXiv:1707.04596* (2017).

[9] Yangchi Chen, Melba M Crawford, and Joydeep Ghosh. 2004. Integrating support vector machines in a hierarchical output space decomposition framework. In *Proceedings of the International Geoscience and Remote Sensing Symposium (IGARSS)*, Vol. 2. IEEE, 949–952.

[10] Caspar J Fall, Atilla Törcsvári, Karim Benzineb, and Gabor Karetka. 2003. Automated categorization in the international patent classification. In *ACM SIGIR Forum*, Vol. 37. ACM, 10–25.

[11] Juan Carlos Gomez and Marie-Francine Moens. 2014. *A Survey of Automated Hierarchical Classification of Patents.* Springer International Publishing, 215–249.

[12] Mattyws F Grawe, Claudia A Martins, and Andreia G Bonfante. 2017. Automated Patent Classification Using Word Embedding. In *Machine Learning and Applications (ICMLA), 2017 16th IEEE International Conference on.* IEEE, 408–411.

[13] Yoon Kim. 2014. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882* (2014).

[14] Aris Kosmopoulos, Ioannis Partalas, Eric Gaussier, Georgios Paliouras, and Ion Androutsopoulos. 2015. Evaluation measures for hierarchical classification: a unified view and novel approaches. *Data Mining and Knowledge Discovery* 29, 3 (2015), 820–865.

[15] Quoc Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *Proceedings of the International Conference on Machine Learning (ICML).* 1188–1196.

[16] Maggie Yundi Li, Liling Tan, Stanley Kok, and Ewa Szymanska. 2018. Unconstrained Product Categorization with Sequence-to-Sequence Models. In *Proceedings of the Workshop on eCommerce (co-located with SIGIR).* 1–6.

[17] Shaobo Li, Jie Hu, Yuxin Cui, and Jianjun Hu. 2018. DeepPatent: patent classification with convolutional neural networks and word embedding. *Scientometrics* 117, 2 (01 Nov 2018), 721–744.

[18] Yukun Ma, Erik Cambria, and Sa Gao. 2016. Label embedding for zero-shot fine-grained named entity typing. In *Proceedings of the International Conference on Computational Linguistics (COLING).* 171–180.

[19] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781* (2013).

[20] Jinseok Nam, Eneldo Loza Mencía, and Johannes Fürnkranz. 2016. All-in text: Learning document, label, and word representations jointly. In *Proceedings of the Conference on Artificial Intelligence (AAAI)*.

[21] Hao Peng, Jianxin Li, Yu He, Yaopeng Liu, Mengjiao Bao, Lihong Wang, Yangqiu Song, and Qiang Yang. 2018. Large-Scale Hierarchical Text Classification with Recursively Regularized Deep Graph-CNN. In *Proceedings of the World Wide Web Conference (WWW)*. International World Wide Web Conferences Steering Committee, 1063–1072.

[22] Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*. 1532–1543.

[23] Florina Piroi, Mihai Lupu, Allan Hanbury, and Veronika Zenz. 2011. CLEF-IP 2011: Retrieval in the Intellectual Property Domain.. In *CLEF (notebook papers/labs/workshop)*.

[24] Julian Risch and Ralf Krestel. 2018. Learning Patent Speak: Investigating Domain-Specific Word Embeddings. In *Proceedings of the International Conference on Digital Information Management (ICDIM)*. 63–68.

[25] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems (NeurIPS)*. 5998–6008.

[26] Oriol Vinyals, Samy Bengio, and Manjunath Kudlur. 2016. Order Matters: Sequence to sequence for sets. In *Proceedings of the International Conference on Learning Representations (ICLR)*. 1–11.

[27] Guoyin Wang, Chunyuan Li, Wenlin Wang, Yizhe Zhang, Dinghan Shen, Xinyuan Zhang, Ricardo Henao, and Lawrence Carin. 2018. Joint Embedding of Words and Labels for Text Classification. *arXiv preprint arXiv:1805.04174* (2018).

[28] Jonatas Wehrmann, Ricardo Cerri, and Rodrigo Barros. 2018. Hierarchical multi-label classification networks. In *Proceedings of the International Conference on Machine Learning (ICML)*. 5075–5084.

[29] Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144* (2016).

[30] Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. 2015. Show, attend and tell: Neural image caption generation with visual attention. In *Proceedings of the International Conference on Machine Learning (ICML)*. 2048–2057.

[31] Gui-Rong Xue, Dikan Xing, Qiang Yang, and Yong Yu. 2008. Deep classification in large-scale text hierarchies. In *Proceedings of the Conference on Research and Development in Information Retrieval (SIGIR)*. ACM, 619–626.

[32] Yan Yan. 2016. Hierarchical Classification with Convolutional Neural Networks for Biomedical Literature. *International Journal of Computer Science and Software Engineering* 5, 4 (2016), 58.

[33] Pengcheng Yang, Xu Sun, Wei Li, Shuming Ma, Wei Wu, and Houfeng Wang. 2018. SGM: Sequence Generation Model for Multi-label Classification. In *Proceedings of the International Conference on Computational Linguistics (COLING)*. 3915–3926.

[34] Dani Yogatama, Daniel Gillick, and Nevena Lazic. 2015. Embedding methods for fine grained entity type classification. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics and the International Joint Conference on Natural Language Processing (ACL and IJCNLP)*, Vol. 2. 291–296.