An LSTM Approach to Patent Classification based on Fixed Hierarchy Vectors

Marawan Shalaby* Jan Stutzki† Matthias Schubert† Stephan Günnemann*

Abstract
Recently, innovative techniques for text processing like Latent Dirichlet Allocation (LDA) and embedding algorithms like Paragraph Vectors (PV) allowed for improved text classification and retrieval methods. Even though these methods can be adjusted to handle different text collections, they do not take advantage of the fixed document structure that is mandatory in many application areas. In this paper, we focus on patent data which mandates a fixed structure. We propose a new classification method which represents documents as Fixed Hierarchy Vectors (FHV), reflecting the document's structure. FHVs represent a document on multiple levels where each level represents the complete document but with a different local context. Furthermore, we sequentialize this representation and classify documents using LSTM-based architectures. Our experiments show that FHVs provide a richer document representation and that sequential classification improves classification performance when classifying patents into the International Patent Classification (IPC) taxonomy.

Keywords: patent classification, word embedding, word2vec, LSTM

1 Introduction

Recently, topic models such as Latent Dirichlet Allocation(LDA) [8] and word embeddings such as word2vec [20] or Paragraph Vectors (PV) [17] demonstrated the capability to map documents into a feature space representing document content independently from the actually used words. Even though these methods might be tuned to any use case, they do not exploit document structure beyond a certain degree. In many document collections, however, all documents are required to have a unique structure. For example, scientific articles appearing at a particular journal or conference usually have to include an abstract, keywords, main sections and a bibliography.

Another very important collection of documents following such a fixed hierarchy are patents. Like publications, patents describe novel methods, technical advances and genuine ideas. However, unlike publications, patents are not formulated to efficiently transfer knowledge but to provide an as broad as possible protection of the contained intellectual property. This often causes large problems when employing standard text processing methods for searching and categorizing patent data. To facilitate patent research, patent offices categorize patents into huge taxonomies like the Cooperative Patent Classification (CPC) or the International Patent Classification (IPC). Manually assigning new patent applications to the correct set of classes within these taxonomies is a time-consuming and expensive process. Thus, automatic patent classification offers the opportunity to speed up the process and relieve experts.

In this paper, we propose a novel classification method for document collections requiring a fixed structure and demonstrate its advantages on the challenging use-case of patent classification. From a technical point of view, our new method is based on learning word embeddings which are specialized on particular parts of a document. Given these specialized embeddings, we consider a document as a hierarchy where parent nodes summarize further partitionings of the document. In particular, the root of the hierarchy describes the complete document, whereas the next levels describe the fixed semantic structure of the document. We denote this representation as fixed hierarchy vectors (FHV) because the partitioning is based on the fixed structure in the analyzed collection. For classification of this new document representations, we propose to sequentialize FHVs using a depth-first-traversal and classify this document representation using a recurrent neural network based on the LSTM [15] architecture. We argue that this traversal is similar to the process of reading from the start to end, but giving the network different level of local context to register various types of information. Our experiments on a large patent corpus demonstrate that FHVs provide a better picture of the document content and that sequential classification yields better performance than vector-based approaches. In summary, the contributions of this paper are:

- A novel document representation for text corpora with a fixed structure.
- A sequential text classifier for classifying the document representations into multiple classes.

*Department of Informatics, Technical University of Munich, Munich, Germany, {*shalaby*, *guennemann*}@in.tum.de
†Institute for Informatics, LMU Munich, Munich, Germany, {*stutzki*, *schubert*}@dbs.ifi.lmu.de

- An evaluation based on the challenging task of patent classification.

The rest of the paper is structured as follows: In Section 2, we survey the state-of-the-art in patent classification and discuss current approaches to text representation. Section 3 specifies the problem setting, describes our novel hierarchical document representation and explains the sequential classification method based on this representation. In Section 4, we show that Fixed Hierarchy Vectors outperform other approaches like paragraph vectors and bag of word approaches (BOW) on a large patent corpus. Finally, we summarize the contributions of this work and discuss directions for future work in Section 5.

2 Related Work and Preliminaries

The most common way to represent text documents for retrieval and classification are Bag Of Words (BOW) approaches where a document is described as a term frequency vector of the contained words [19]. The most prominent BOW measure [2] is Term Frequency Inverse Document Frequency (TF/IDF). A more refined TF/IDF-like method is the Okapi Best Match 25 (BM25) [23] scoring function. We will compare to BM25 in our experiments due to its observed superior performance for our use case of patent classification. A drawback of weighted term frequency vectors is that documents about the same topics might not be recognized as similar if the used terms do not overlap enough. Thus, topic models were introduced, mapping words to a latent representation space describing possible topics. Currently, the most prominent topic modeling approach is Latent Dirichlet allocation (LDA) [8]. LDA is a generative model that uses sampling techniques to iteratively refine topics in a way that the final model would produce a similar term distribution to a training document for identical topic distributions. Different extensions to LDA allow online processing [1] and hierarchical topics [21]. Although topic models describe the connection between words and topics, they do not consider the local context in which a term is used. Bengio et al. [3] propose a novel approach considering local context and the order in which terms occur in a text. In this approach, a neural network learns a low-dimensional token representation and predicts the next token within a window around each term based on the learned representation. This work is built upon by Mikolov et. al [20], who introduces word2vec where the network architecture is simplified by only learning the token representation as a lower dimensional vector which is also referred to as an embedding. Word2vec includes two approaches: the skip-gram model where the context is predicted based on the token and the Continuous Bag of Words model (CBOW) where the current token is predicted based on its surrounding tokens. While skip-grams provide better results in semantic tasks, the CBOW architecture is faster in training and slightly more performant in syntactic benchmarks.

Le et al. [17] extend word2vec by introducing Paragraph Vectors (PV). PV add an additional vector space representing entire paragraphs. This vector space is learned jointly with the word vectors of the tokens in the paragraphs. Analogously to CBOW and Skip-gram for word2vec, PV include two strategies: PV Distributed Memory (PV-DM) uses a sliding window of context tokens in the current paragraph and tries to predict the target word. The second strategy, PV Distributed Bag of Words (PV-DBOW) uses one token in the paragraph as input and tries to predict the surrounding words in the current window. In [12] an extension to PV is proposed which builds on the structure within a document. The method learns a single model from training instances from all structural context levels. The resulting vectors for each context level are then concatenated to represent the document. Evaluation on a binary classification task showed degrading performance with each added level of detail and a significant increase in training time. Though this method is similar to the FHV being proposed in this paper, FHVs use different models for different parts of the document and enable a sequential classification method leading to an increased performance when adding additional levels.

After describing general methods for text representation, we will now survey the state-of-the-art for patent classification. Benzineb et al. [5] provide an overview of the current state of the art in the field of automatic patent classification, stating that SVCs and neural networks(NN) represent the current best practice solutions. A comparison of patent classification methods Balanced Winnow [18] and linear Support Vector Classifier (SVC) [10, 7] draws the conclusion that SVCs consistently outperform Balanced Winnow. [7] proposes to use Logarithmic Term Count (LTC) instead of TF/IDF to reduce the effect of large differences in term frequency. The authors of [11] focus on the benefits of Mutual Information to classify patents into the United States Patent Classification (USPC) System. They follow a BOW approach utilizing a linear soft margin SVC and apply Mutual Information to the text. Automated patent classification into the IPC is examined in [13]. Naive Bayes and SVC are compared with SVCs providing similar or superior results using a customized success criterion. In [16] the authors investigated which part of a patent provides the best results for classification. Results show that the more text is available the better SVCs perform.

Beside full-text approaches, methods utilizing the meta data are also explored. One method uses the addresses of involved parties to generate a spatial probability distribution of patent classes [24]. Though this method improves classification performance, it should be used in combination with text classification.

The method proposed in this paper is based on classifying text as a sequence of vectors in a word embedding vector space. To classify sequential inputs, we rely on recurrent neural networks. Recurrent Neural Networks (RNNs) provide an even richer dynamic representation of functions, by including weighted self loops within their hidden layers, allowing them to operate on sequences of data instances of arbitrary lengths. Theoretically, RNNs are able to approximate any sequence to sequence mapping $h : (x_1, x_2, .., x_T) \mapsto (y_1, y_2, .., y_U)$ where $U \leq T$ [14]. In practice however, many RNN architectures face major difficulties in training over long sequences of time steps due to the vanishing/exploding gradient problem [4] caused by the weights on the recurrent connections. Long Short Term Memory (LSTM) models [15] were proposed to fix this issue.

3 Fixed Hierarchy Vectors

3.1 Problem Setting We consider a text corpus $D = \{d_1, \ldots, d_n\}$ where d_i is a structured document. In this paper, each document d_i is described as a sequence of tokens $(t_1, \ldots, t_m)$ such as words, shingles or other textual primitives. The structure of document $d_i \in D$ is given as a tuple $d_i = (p_1, \ldots, p_l)$ of partitions p_j. For example, in our use case of patent classification each patent is represented as a tuple of three partitions $(abstract, description, claims)$. Each partition $p_j = (t_k, \ldots, t_l)$ describes a subsequence of document d_i starting with index k and ending with index l. Since the structure is considered to be a complete and disjoint partitioning, the next partition p_{j+1} must start with $l+1$. Note that each part p_j can have a structure of its own and thus, we can build hierarchical structures. We operate in a multi-label classification problem setting, where each document $d_i \in D$ can belong to multiple classes $c_j \in C$ where C denotes the set of all classes and $class(d) = \{c_j, .., c_k\}$ denotes the set of all classes that document d belongs to. In our use case, we use the first three levels of the international patent classification (IPC) comprising 8 sections (level 1), 244 classes (level 2) and 940 subclasses (level 3). Our task is to train a function $f : D \rightarrow 2^C$ to map a document d to the correct set of classes $class(d)$.

Our solution to classifying structured documents involves two steps. In the first step, we propose an unsupervised representation learning method which is based on the recursive partitioning provided by the document structure. The result is a hierarchy of embedded feature vectors, each describing a partition in the document structure. In the second step, we traverse the hierarchy of embedded vectors to generate a sequence which is used to train a recurrent neural network, in our case an LSTM [15], for document classification.

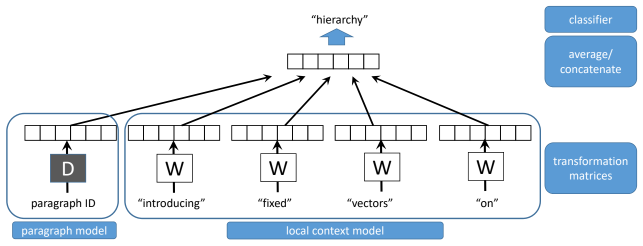

Figure 1: An overview of the employed PV-DM model [17]. The paragraph matrix D is trained to provide the global context for each token window representing the local context.

3.2 Fixed Hierarchy Vectors In this section, we describe the transformation of a text document into a representation which is more suitable to capture its contents.

FHV strongly rely on the PV model due to its ability to learn embeddings based on any type of content. Therefore, we will give a short summary on computing the PV distributed memory (PV-DM) model as described in [17]. The goal of PV-DM is to learn a mapping $\Phi(P)$ for an input paragraph P to a q-dimensional vector space describing P's contents. Formally, $P = (t_1, \ldots, t_m)$ is a sequence of m tokens representing either words, shingles or other textual primitives. The mapping is based on two linear transformations D and W. Whereas D maps the paragraph id pid to the target space, W maps the tokens in the local context of each token in P to the target space. In particular, the local context for token $t_i \in P$ is represented by its local context $t_{i,context} = \{t_{i-w}, ..., t_{i-2}, t_{i-1}, t_{i+1}, t_{i+2}, ..., t_{i+w}\}$ surrounding t_i with the window size w. Since every token $t_i \in P$ and the given paragraph id provide a q-dimensional output vector, all output vectors are either concatenated or averaged into a single q-dimensional vector. To learn both matrices, the PV model trains a two layer architecture which is depicted in Figure 1. The idea of training is to maximize the average log likelihood to predict a token t_i based on its context $t_{i,context}$ and the paragraph id pid it occurs in:

$$(3.1) \qquad \frac{1}{m} \sum_{i=w}^{m-w} \log p(t_i | t_{i,context}, pid)$$

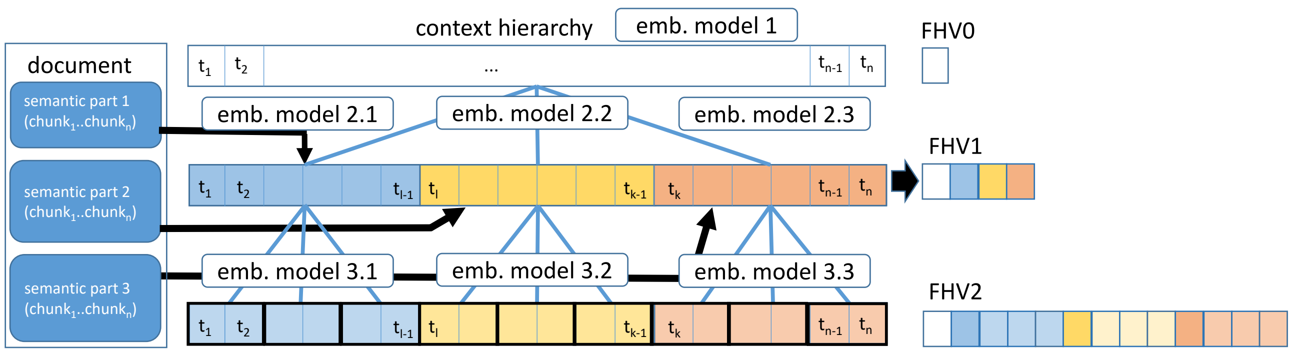

Figure 2: An example for deriving an FHV representation for a document having three partitions (left). In the middle, the context hierarchy of the document is shown. The right side depicts sequential FHV representations. Whereas FHV0 uses the complete document as context and summarizes the document as one embedded vector, FHV2 represents the document as sequence of embedded vectors combining context information on different levels of the hierarchy.

Now, $p(t_i|t_{i,context}, pid)$ is trained as a softmax classifier which is based on a linear prediction function:

$$p(t_i|t_{i,context}, pid) = \frac{e^{y_{t_i}}}{\sum_j e^{y_j}} \tag{3.2}$$

where $y_{t_i} = b + U\Phi(t_{i,context}, pid))$. Let us note that Φ can be applied to $t_{i,context}$ as well as the complete paragraph P because adding the tokens of the complete paragraph only adds additional vectors for the average function aggregating the final result. Training is performed with Stochastic Gradient Descent (SGD). As in word2vec, negative sampling is applied. For mapping a novel paragraph, unknown paragraph ids have to be integrated into the paragraph matrix D. Thus, new columns have to be added to D and afterwards, D is optimized again using gradient descent while freezing all other parameters h, U, and W.

After reviewing PVs, we will now introduce our new method fixed hierarchy vectors (FHV). As mentioned above, a document d_i is represented as an ordered tuple $(p_1, \ldots, p_l)$ of partitions. Each partition p_i consists of a sequence of tokens $(t_1, \ldots, t_m)$. A token can either be a term or a sequence of characters, e.g a shingle. A key task of learning a good text representation is to consider the right context to interpret each token. Whereas word2vec showed that the local context of surrounding words is important, other approaches, e.g. paragraph vectors, showed that this very local context is often not enough to completely capture the meaning of each token. Thus, in our new approach, we follow the idea of combining text representations for various levels of locality from surrounding tokens up to the complete document.

In the following, we will represent each document d_i as a context hierarchy $H(d_i)$. Formally, $H(d_i)$ is a tree where each node $N_{i,j}$ describes a part of the document, i.e. some token subsequence $(t_i, \ldots, t_j)$ starting at token t_i and ending at token t_j. The children of $N_{i,j}$ represent a complete and disjoint partitioning of $(t_i, \ldots, t_j)$. In other words, l children nodes define $l-1$ split indices between i and j. Thus, each level of $H(d_i)$ always contains all tokens in d_i, but lower levels split the token sequence of d_i into successively more partitions.

We generate $H(d_i)$ as follows: The root level represents the complete document d_i. The next level is denoted as semantic level because we split d_i along the fixed semantic partitioning $(p_1, \ldots, p_l)$ being characteristic to the document collection. Finally, on the third level, we split each partition p_i into a fixed number of chunks k_i. Chunking is done by considering the complete number of tokens m in partition p_i and evenly split p_i after $\lceil \frac{m}{k_i} \rceil$tokens. Note that each partition might have a varying number of chunks which is selected based on the average length of the partition. For example, the description part of a patent will be split into more chunks than the abstract part which naturally contains far less tokens.

After defining the context hierarchy, we describe each node in the hierarchy by a meaningful embedding vector. To take full advantage of the variety of contexts, we need a feature transformation taking the given level of context into account. Since paragraphs in PV [17] are not strictly defined but can be applied to any partitioning of text like document, sentence, chunk or section, PV perfectly complement the document hierarchy proposed above. Therefore, we employ the PV model to learn an individual feature transformation for the nodes in the context hierarchy $H(d_i)$. In particular, we train a model for the root level and each node on the semantic level. For the chunk level, we train a single model for all the children of each node on the semantic level. For example, we train one model over all chunks in the description part. Thus, the number of trained PV models is $1 + 2 \cdot l$, i.e. one for the root, l for the semantic partitions and l for the chunk level. In our patent use case, we train 7 PV models given that there are 3 semantic partitions. In our experiments, PV-DM outperformed PV-DBOW in most of the cases and thus, we use the PV-DM model for learning FHVs. Within the PV-DM model, we combine intermediate results by taking the average due to its better performance and faster runtimes compared to concantenation.

For inference on a new document $\hat{d}$, we first partition $\hat{d}$ into the context hierarchy $H(\hat{d})$. Afterwards, we apply the PV model for each node of the hierarchy to map each node to the q-dimensional embedding vector where q corresponds to the dimensionality of the latent space in all the PV models. Thus, we derive a single vector for each node in the context hierarchy $H(\hat{d})$. An overview on the complete document representation is depicted in Figure 2.

3.3 Using LSTMs for FHV classification The simplest way to do document classification based on FHVs is to flatten the hierarchy by concatenating the vectors representing its nodes. Let us note that this is possible because the number of nodes in the context hierarchy is exactly the same for each document. However, it turned out that this simple solution does not provide the best results for text classification as can been seen in our experimental section Section 4.

Understanding a complicated text document for a human is often a timely process. To capture the contents of a complex document as in our use case of patents, people often read multiple times over the same document. Whereas the first read has the purpose of getting a general overview, subsequent reads aim to get a deeper understanding of the details. The reason for this behavior is that some of the details within the document cannot be judged correctly without knowing the complete context. Another aspect which is natural to understanding text is sequential parsing. Usually a document is read from the beginning and often sections within a document cannot be understood correctly, without knowing the contents of previous sections. For example, in a research paper, the method description usually depends on nomenclature and definitions being described earlier in the document. To build a classification scheme which mimics both effects, we propose to traverse the FHV hierarchy in a depth-first order and thus, generate a sequence of q-dimensional embedding vectors. Thus, the complete sequence has a length of $1 + l + \sum_{i=0}^{l} k_i$ vectors. The sequence starts with a vector describing the complete document at root level and thus, represents a quick overview on everything contained. Then, we continue by examining the semantic partitioning by considering an overview over each partition on the second level and then, descend to the chunk levels for exploring the details before proceeding to the next semantic partition.

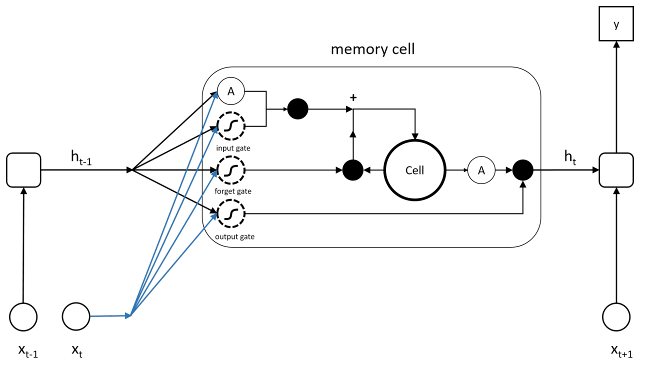

Figure 3: The LSTM architecture for the input x_t in step t and the output h_t.

For classifying FHVs, it is extremely important for the employed classification architecture not to have a fixed time horizon but can rather keep any relevant information previously seen while parsing the document description. Thus, all sequential classification schemes using a fixed-order Markov property are not suitable to exploit FHVs.

A method providing the wanted property is Long Short Term Memory (LSTM) recurrent neural networks[15]. Though we cannot give an in-depth description on how the used LSTM [15] architecture works, we shortly explain how this architecture provides the required properties. An overview of the LSTM architecture can be found in figure 3. In contrast to back-propagation through time networks, LSTMs replace the hidden neurons with a more complex structure called a memory cell containing a hidden state and 3 sigmoid multiplicative gates called the input, output and forget gates. These gates dictate respectively whether the cell should accept input from the previous time steps, whether it should output its current state to the next timestep, and whether it should keep its current state. This setup leads to an RNN that is able to operate on very long time series with negligible attenuation of the gradient. For our use case, this means that relevant local context information can be stored while parsing the FHVs if it is relevant to determine the correct classes for a document.

We experimented with several variations of this basic architectures but a single LSTM layer already showed a good trade-off between prediction quality and model complexity. The evaluated architectures as well as hyperparameter tuning of the LSTM classifiers are described in Section 4.

To implement multi-label classification, we built the output layer with a dimensionality of C. Thus, the clasifier performed a one-vs-rest classification for each class and the document was predicted to belong to a class if the corresponding output surpased a given threshold τ. In our experiments, we used a threshold of $\tau = 0.5$.

4 Experimental Evaluation

4.1 Experimental setting The dataset used throughout this paper is composed of 1,688,202 patents published between 2006 and 2015. The patents span all 8 top-level IPC-Sections (A to H). We split the dataset into 1,286,325 documents for training and 401,877 documents for testing. The documents also span 244 IPC-Classes and 940 IPC-Subclasses. Most documents are concentrated in the G and H sections, representing 35.7% and 29.9% respectively, while sections D and E contain only 0.6% and 2.4% of all documents. We observe the greatest overlap of the two IPC-Sections G and H with around 4% of all documents having both of those sections.

To evaluate our classifiers, we use the standard F1 Micro and F1 Macro measures commonly used in text classification tasks. Micro-averaging sums up the classification decisions of all instances (whether an instance is a true positive, false negative, etc..) regardless of class, then computes the F1 score based on this aggregate sum. This helps to give a general picture of how effective the classifier is in general at identifying the correct labels for documents. Macro-averaging on the other hand computes the F1 score independently for every class, then averages the F1 scores for all classes to arrive at the F1 Macro score. This can give us a more complete picture of how the classifier fares when we have an unbalanced distribution of the number of documents for each class whereas classes with a large number of documents can dominate smaller classes in micro-averaging. We also use two additional measures geared towards multi-label classification: Coverage Error and Top 3 percentage. Coverage error refers to how far we need to go down a ranked list of labels on average in order to account for all the true positive labels. It is calculated as follows:

$$\text{(4.3) Coverage Error} = \frac{1}{N}\sum_{i=1}^{N} \max_{y_j \in Y[i]} rank(y_j) - 1$$

While Top 3 calculates the percentage of correct labels that are within the top 3 scores.

Table 1: Section Classification Results

Pipeline	Cov.Er.	Mic.F1	Mac.F1	Top3
SVC&X^2	1.696	0.734	0.671	0.924
SVC&LDA	1.875	0.617	0.401	0.896
SVC&FHV_0	1.733	0.663	0.569	0.923
MLP&X^2	1.425	0.791	0.704	0.972
RNN&X^2	1.429	0.786	0.693	0.973
MLP&LDA	1.462	0.776	0.689	0.967
RNN&LDA	1.482	0.763	0.672	0.965
MLP&FHV_0	1.414	0.793	0.736	0.975
RNN&FHV_0	1.410	0.798	0.747	0.975
MLP&FHV_1	1.380	0.814	0.766	0.978
RNN&FHV_1	1.368	0.821	0.776	0.980
MLP&FHV_2	1.374	0.818	0.772	0.979
RNN&FHV_2	**1.355**	**0.828**	**0.787**	**0.982**

For the paragraph vectors, we use the highly efficient multi-threaded Gensim [22] Python implementation. The training of the PV models ran on a 32-core machine with 380GB of RAM using multiple concurrent pipelines. Creating the paragraph vectors for all different nodes in the hierarchy took around 2 days to complete. For MLPs and RNN, we used the Keras learning library [9] and experiments are performed on a 64GB RAM machine with an Nvidia Titan X. For reproducibility, the code is available online [1].

4.2 Classification Results For a concise evaluation of the classification performance of our method, we compare various combinations of classifiers and vector representations. We compare the following classification methods: Linear Support Vector Classifier (SVC), Multi-Layer Perceptrons (MLP) and RNNs. SVC uses a One-vs-Rest strategy for training a linear kernel. The parameters for SVC are chosen using a validation set. Parameters and architectures for MLP and RNN are the result of our hyper parameter evaluation which is described in Section 4.3 in more detail.

To evaluate the document representations, we employ BM25 [19] on unigrams and bigrams, LDA, PV and FHV. BM25 is selected as experiments on a smaller training/validation set proved it to be superior to other BOW representations. Feature selection with χ^2 is performed on the BM25 vector space to retain the top 10,000 features (X^2). The parameter evaluation for LDA leads to a dimensionality of 1000 topics which provided the best results. With MLP and RNN outperforming the SVC on PV, LDA and X^2 by a wide

[1]The code is available at
`https://github.com/marawanokasha/fhv_paper_code`

Table 2: Class Classification Results

Pipeline	Cov.Er.	Mic.F1	Mac.F1	Top3
SVC&X^2	6.359	0.630	0.178	0.735
SVC&LDA	48.591	0.417	0.032	0.611
SVC&FHV_0	5.074	0.542	0.130	0.665
MLP&X^2	2.866	0.682	0.206	0.861
RNN&X^2	2.927	0.665	0.176	0.859
MLP&LDA	3.440	0.640	0.155	0.827
RNN&LDA	3.544	0.616	0.154	0.816
MLP&FHV_0	2.791	0.663	0.208	0.857
RNN&FHV_0	2.698	0.684	0.225	0.866
MLP&FHV_1	2.535	0.703	0.240	0.877
RNN&FHV_1	2.438	0.712	**0.255**	0.885
MLP&FHV_2	2.501	0.710	0.245	0.882
RNN&FHV_2	**2.410**	**0.721**	0.251	**0.889**

Table 3: Subclass Classification Results

Pipeline	Cov.Er.	Mic.F1	Mac.F1	Top3
SVC&X^2	16.322	0.507	0.130	0.488
SVC&LDA	165.311	0.266	0.014	0.161
SVC&FHV_0	15.784	0.423	0.093	0.325
MLP&X^2	7.096	0.567	0.168	0.753
RNN&X^2	7.381	0.515	0.107	0.741
MLP&LDA	9.510	0.506	0.102	0.696
RNN&LDA	9.588	0.473	0.103	0.689
MLP&FHV_0	6.818	0.526	0.167	0.737
RNN&FHV_0	6.531	0.559	0.186	0.749
MLP&FHV_1	5.860	0.583	0.201	0.768
RNN&FHV_1	5.677	0.605	0.223	0.782
MLP&FHV_2	5.728	0.590	0.206	0.775
RNN&FHV_2	**5.482**	**0.612**	**0.215**	**0.789**

margin, we do not perform further experiments with the other representations for SVC. For our FHV method, we distinguish the semantic hierarchy levels $l \in [0, 1, 2]$ of the input data FHV_l in our classification experiments as follows:

- FHV_0: we use only the root level of $H(d)$. This method corresponds to the standard approach when using PVs.
- FHV_1: we use the root and semantic levels of $H(d)$.
- FHV_2: we use all three levels in $H(d)$. In total, we use 30 chunks: 3 chunks for the abstract, 23 chunks for the description and 4 chunks for the claims.

All of our FHV experiments are conducted on a vector space with 200 dimensions and the embeddings are trained for 8 epochs with a dictionary size of about 400,000 terms remaining after imposing a minimum word count threshold of 100 occurrences. We use RMSprop as the learning algorithm and a batch size of 4096 with the exception of experiments with FHV_2 data where, due to memory constraints, we decreased the batch size to 2048.

For regularization of the neural networks, we use early stopping with a patience of 15 epochs and a minimum required decrease of 1^{-5} in validation loss. Dropout is used throughout the network with $p = 0.5$. The same values are also used for the RNNs.

Tables 1, 2 and 3 show the results of our experiments for the section, class and subclass levels of the IPC patent classification. We can see a consistent advantage for RNNs over MLPs when it comes to FHV representations and a consistent increase in performance for both models as we add more levels.

Table 4: One Model Results for LSTMs

Pipeline	Cov.Er.	Mic.F1	Mac.F1	Top3
Sec.&FHV_0	1.585	0.697	0.605	0.950
Sec.&FHV_1	1.465	0.771	0.709	0.968
Cl. &FHV_0	3.550	0.558	0.119	0.807
Cl.&FHV_1	3.055	0.636	0.172	0.837
Sub.&FHV_0	10.567	0.399	0.059	0.667
Sub.&FHV_1	8.036	0.502	0.120	0.714

For the SVC classifier, FHV_0 lags behind X^2 in F1 Micro, F1 Macro and Top 3, but that can be explained by the fact that the lower dimensional representation employed by FHV is not as amenable to linear separation as the high dimensional bag of words format, and may require a non-linear discriminator. LDA is another representation that benefits greatly from the non-linearities of MLP and RNN even though it is unable to beat either X^2 or FHV.

We also test whether the use of independent models for the varying nodes in the tree as we have explained in Section 3 is justified. Thus, we experimented with adding the text in all of the nodes in the tree to one model and train this monolithic model using all the documents and their partitions. As Table 4 shows, the results of this experiment reaffirmed that the use of independent models for independent contexts is necessary to achieve good results as the results of the monolithic model lag far behind those of the independent models.

4.3 Hyper Parameter Evaluation For MLPs, we selected the parameters using a random search [6] procedure on a validation set over a grid of parameters for layer size (100 to 2000), number of hidden layers

Table 5: RNN Stacking for sections

Model	Cov.Er.	Mic.F1	Mac.F1	Top3
1-l.&$FHV0$	1.410	0.798	0.747	0.975
2-l.&FHV_0	1.390	0.808	0.761	0.978
3-l.&FHV_0	1.391	0.808	0.761	0.978
1-l.&FHV_1	1.368	0.821	0.776	0.980
2-l.&FHV_1	1.362	0.823	0.780	0.981
3-l.&FHV_1	1.360	0.824	0.781	0.981

(1, 2 or 3) and hidden layer activation functions (relu, sigmoid and tanh). We report the numbers for the best performing combination of each experiment. For RNNs which require much longer to train, we did a grid search over a more constrained set of parameters. We used tanh as the activation function and only did grid search on the hidden layer size where we found 1000 to be the best performing across our experiments.

In the previous section, we used only a single layer RNN, however adding stacking to our RNN classifiers can also lead to performance gains. Table 5 shows the results of adding RNN stacking for the same model for sections and it clearly shows an advantage for adding a second stacked RNN layer. Adding a third layer however leads to additional model complexity and runtime duration, but no significant performance improvements.

We also tested the effect of training epochs for learning paragraph vectors and the effect the number of epochs has on the performance of our model. As figure 4 shows, running paragraph vectors for around 3 epochs is more than enough to produce good results. When training for more epochs, the improvements diminish more and mores. This is of course dependent on the number of training instances in the dataset because when we tested with a smaller dataset, it took more epochs to reach a point of diminishing returns.

Finally, we tried adding a convolutional layer before the LSTM layer as a way of finding out if more descriptive features can be extracted from the paragraph vectors being fed into the LSTM network, however we discovered that adding the convolutional layer did not lead to any significant increases in performance. Adding a max pooling layer after the convolutional layer always led to a worse performance.

5 Conclusions

In this paper, we present a new approach to classify documents with an inherent structure in multi-class settings. Our method uses the inherent structure to derive a hierarchical description of the document. Whereas each level in this hierarchy represents a different level of summarization, the children of each node represent a sequential partitioning of the parent node. For each node in the hierarchy, we train a paragraph vector model mapping the corresponding text in the document to a lower dimensional representation vector. To classify a document, we propose a depth-first traversal through the hierarchy resulting in a sequential representation and use an LSTM for classification. The results indicate that our new method can beat state-of-the-art approaches in the challenging use case of patent classification. Furthermore, our experiments indicate that sequential processing using an LSTM shows a better performance than general multilayer perceptrons on a concatenation of the complete sequence. One of the limitations of our approach is that it requires a fixed hierarchy for all documents in the corpus, which is probably not the case for all document corpora. A future direction for our work is to figure out if imposing a synthetic hierarchy for a document (e.g. by using chunking starting from the first level in the tree as opposed to just in the leaf nodes) would lead to improved results as well. Additionally, we plan to integrate both steps into a single architecture to allow joint optimization. Furthermore, we want to extend the LSTM architecture to integrate more recent improvements like attention mechanisms which consider the hierarchical structure of the document.

Acknowledgments. Jan Stutzki was supported by the Audi Electronics Venture GmbH. Stephan Günnemann was supported by the German Research Foundation, Emmy Noether grant GU 1409/2-1, and by the Technical University of Munich - Institute for Advanced Study, funded by the German Excellence Initiative and the European Union Seventh Framework Programme under grant agreement no 291763, co-funded by the European Union.

References

[1] L. AlSumait, D. Barbará, and C. Domeniconi, *On-line lda: Adaptive topic models for mining text streams with applications to topic detection and tracking.*, in 2008 Eighth IEEE International Conference on Data Mining, IEEE Computer Society, Dec 2008, pp. 3–12.

[2] J. Beel, B. Gipp, S. Langer, and C. Breitinger, *Research-paper recommender systems: a literature survey*, International Journal on Digital Libraries, 17 (2016), pp. 305–338.

[3] Y. Bengio, R. Ducharme, P. Vincent, and C. Jauvin, *A neural probabilistic language model*, Journal of Machine Learning Research, 3 (2003), pp. 1137–1155.

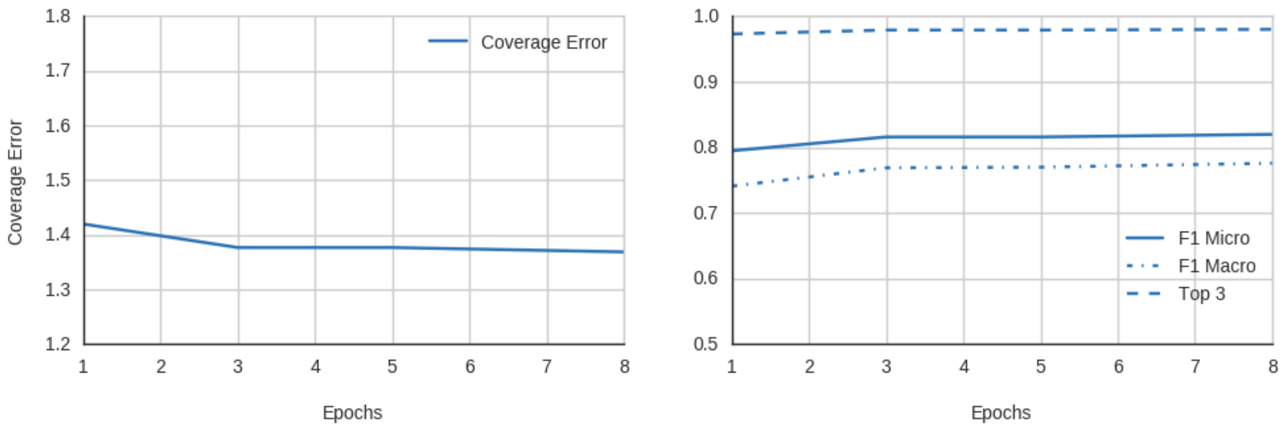

Figure 4: Progress over epochs for sections on the same RNN model for FHV1.

[4] Y. Bengio, P. Simard, and P. Frasconi, *Learning long-term dependencies with gradient descent is difficult*, IEEE Transactions on Neural Networks, 5 (1994), pp. 157–166.

[5] K. Benzineb and J. Guyot, *Automated patent classification*, in Current challenges in patent information retrieval, Springer, 2011, pp. 239–261.

[6] J. Bergstra and Y. Bengio, *Random search for hyper-parameter optimization*, Journal of Machine Learning Research, 13 (2012), pp. 281–305.

[7] K. Beuls, B. Pflugfelder, and A. Hanbury, *Comparative analysis of balanced winnow and svm in large scale patent categorization*, in Proceedings of Dutch-Belgian Information Retrieval Workshop (DIR), 2010, pp. 8–15.

[8] D. M. Blei, A. Y. Ng, and M. I. Jordan, *Latent dirichlet allocation*, Journal of Machine Learning Research, 3 (2003), pp. 993–1022.

[9] F. Chollet, *Keras*. `https://github.com/fchollet/keras`, 2015.

[10] C. Cortes and V. Vapnik, *Support-vector networks*, Machine learning, 20 (1995), pp. 273–297.

[11] I. Costantea, R. I. Boţ, and G. Wanka, *Patent document classification based on mutual information feature selection*, Technische Universität Chemnitz. Fakultät für Mathematik, 2004.

[12] L. Elmer, *Hierarchical paragraph vectors*, master's thesis, Eidgenössische Technische Hochschule Zürich, 2015.

[13] C. J. Fall, A. Törcsvári, K. Benzineb, and G. Karetka, *Automated categorization in the international patent classification*, SIGIR Forum, 37 (2003), pp. 10–25.

[14] A. Graves, *Supervised sequence labelling*, Springer, 2012.

[15] S. Hochreiter and J. Schmidhuber, *Long short-term memory*, Neural Computation, 9 (1997), pp. 1735–1780.

[16] L. S. Larkey, *A patent search and classification system*, in Proceedings of the Fourth ACM conference on Digital Libraries, August 11-14, 1999, Berkeley, CA, USA, ACM, 1999, pp. 179–187.

[17] Q. V. Le and T. Mikolov, *Distributed representations of sentences and documents.*, in ICML, vol. 14, 2014, pp. 1188–1196.

[18] N. Littlestone, *Learning quickly when irrelevant attributes abound: A new linear-threshold algorithm*, Machine Learning, 2 (1987), pp. 285–318.

[19] C. D. Manning, P. Raghavan, and H. Schütze, *Introduction to Information Retrieval*, Cambridge University Press, 2008.

[20] T. Mikolov, K. Chen, G. Corrado, and J. Dean, *Efficient estimation of word representations in vector space*, arXiv preprint arXiv:1301.3781, (2013).

[21] A. J. Perotte, F. Wood, N. Elhadad, and N. Bartlett, *Hierarchically supervised latent dirichlet allocation*, in Advances in Neural Information Processing Systems, 2011, pp. 2609–2617.

[22] R. Řehůřek and P. Sojka, *Software Framework for Topic Modelling with Large Corpora*, in Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks, Valletta, Malta, May 2010, ELRA, pp. 45–50.

[23] S. E. Robertson, S. Walker, S. Jones, M. M. Hancock-Beaulieu, M. Gatford, et al., *Okapi at trec-3*, NIST Special Publication SP, 109 (1995), p. 109.

[24] J. Stutzki and M. Schubert, *Geodata supported classification of patent applications*, in Proceedings of the Third International ACM SIGMOD Workshop on Managing and Mining Enriched Geo-Spatial Data, ACM, 2016, p. 4.