MEXN: Multi-Stage Extraction Network for Patent Document Classification

Juho Bai 1Ø, Inwook Shim 2•* and Seog Park 1•*

1 Department of Computer Science and Engineering, Sogang University, Mapo-gu, Seoul 04107, Korea; juho@sogang.ac.kr
2 The Ground Autonomy Laboratory, Agency for Defense Development, Daejeon 34186, Korea
* Correspondence: iwshim@add.re.kr (I.S.); spark@sogang.ac.kr (S.P.)

Received: 3 August 2020; Accepted: 4 September 2020; Published: 8 September 2020

Abstract: The patent document has different content for each paragraph, and the length of the document is also very long. Moreover, patent documents are classified hierarchically as multi-labels. Many works have employed deep neural architectures to classify the patent documents. Traditional document classification methods have not well represented the characteristics of entire patent document contents because they usually require a fixed input length. To address this issue, we propose a neural network-based document classification for patent documents by designing a novel multi-stage feature extraction network (MEXN), which comprise of paragraphs encoder and summarizer for all paragraphs. MEXN features analysis of the whole documents hierarchically and providing multi-labels outputs. Furthermore, MEXN preserves computing performance marginally increase. We demonstrate that the proposed method outperforms current state-of-the-art models in patent document classification tasks with multi-label classification experiments for USPD datasets.

Keywords: document classification; automatic patent classification system; attention network

1. Introduction

Recently, neural network methods have various influences on the natural language process, including automatic document classification problems. A typical example of automatic document classification used in real life is the automatic patent classification (APC), which is a useful way to mitigate the enormous cost of analyzing patent documents as the number of patent documents increases. With the dramatic rise in the adoption of deep learning in APC in the past few years, recent studies [1–5] tried to learn the characteristics of the patent documents (e.g., novelty and inventive step) from the keywords itself the patent documents itself. In the early approaches, a simple convolution neural network (CNN) has been used to learn the semantic meaning of the documents [1]. Then, integrated model approaches [4–6] have been introduced for a deeper understanding of the semantic meaning of the documents, which has frequently combined CNN with recurrent neural networks (RNN [7]). Jointly training their combined architecture discriminatively allows one to leverage the expressive power of deep neural networks. Furthermore, a self-attention mechanism is also adopted to capture long-range interactions of the patent documents [3]. These methods show that the general document classification model can be applied to the patent document classification work even if the entire contents of the patent are not used.

Despite the success of recent works, APC is still challenging as they have designed deep network architectures without deeply understanding the contents of novelty and inventive step, which are typical characteristics of the patent documents. These two attributes are determined by a series of claims and descriptions, and each paragraph contains the contents of each paragraph. The combination and relationship of the patent contents plays a essential role in generating the semantic meaning of novelty.
and inventive step. Therefore, in order to thoroughly analyze a patent document, it is required to design the network architecture in consideration of analyzing the “whole” document. Moreover, recent studies have also failed to show various metrics for evaluating multi-label classification performance, which is one of the core characteristics of patent documents.

In this paper, we aim to better understand the patent document by hierarchically analyzing the structure of the “whole” patent document with marginally increasing the computational cost, taking into account the novelty and inventive step both, thereby obtaining well-balanced documents features that embrace all paragraphs (patent claims and their descriptions). We design the whole process in three stages. First, we divide the input document to fixed-length paragraphs and extract features with weight sharing networks. In the second stage, the paragraph features are summarized into a document feature by using the attention mechanism. Finally, we compute the weight matrix with hierarchical category information of the patent.

**Contributions.** Our main contributions are summarized as follows:

- MEXN is a novel document classification network for patent documents that analyzes structural configuration and entire contents through a multi-stage feature aggregation.
- We introduce a parallel and hierarchical network design to improve classification performance and reduce the computational cost.
- Investigations on the extendability of MEXN are conducted by proposing a structural derivative. MEXN has a flexible structure in which various existing methods can be used on each stage.
- We analyze and evaluate MEXN through extensive experiments on multiple benchmarks. The experiments show that our network achieves the great improvement of the patent document classification task.

2. Related Work

2.1. **Rule-Based and Machine Learning (Data-Driven) Based Methods**

There are various studies before deep learning approaches in APC was successfully applied. Expert system [8] is one of the excellent models of classical document classification methods. This method has been used in various studies until recently. Many approaches [9–11] try to obtain high-quality classification results using the expert system. However, it cannot avoid the fact that it is highly dependent on the quality of human experts.

To alleviate this problem, statistical analysis models have begun to be introduced in document classification methods [9,12–14]. These methods combine hand-crafted document analysis with conventional machine learning models such as TF-IDF [15] and SVM [16]. These methods show excellent performance given the words and their combinations are sufficiently distinct. However, it cannot distinguish subtle differences between similar contexts. In this paper, we adopt domain knowledge of expert systems in the label prediction process by utilizing the multi-label characteristics of the patent documents. In other words, the top-label information can guide a classifier network when it trains the sub-label classifier, and vice-versa.

2.2. **Deep Learning Methods on Text Classification**

A recent study of text classification methods [17] shows using supervised learning approaches with CNN, RNN and Transformer methods, and unsupervised learning approaches. Specifically, Transformer [18–22] based studies show good performance in text classification work. Transformer-XL [20] is based on the AR method, so it computes sequentially. Therefore, it is difficult to use when the entire information should be handled simultaneously, such as document classification. XLNet [21] is based on the AR method but adapts the AE method to show excellent performance. However, since this method considers the AE method together, there is a limit to the computational cost with long-length inputs. Longformer [22] uses transformer encoders with a sliding window method to take long-length input. Although the method with a sliding window approach can handle long-term
input, it is not suitable for APC due to the possibility of losing important information because of data loss.

2.3. Deep Learning Methods on APC

Deep learning approaches have driven tremendous successes in the patent classification task. However, the challenges of efficiently abstracting quite a long patent document have impeded natural transplantation on this task. DeepPatent [2] embeds each word independently using word vector embedding method [23] and sentence-level CNN [1]. DeepPatent classifies the document by considering the relationship between embedded words. Because this simple relationship between word features ignores long-range connectivities, RNN-Patent [24] adopts the bi-directional recurrent model [25] to broaden the receptive field of the network. Furthermore, PatentBERT [3] exploits the global structure of the patent documents by utilizing self-attention mechanisms [18,19], which are used to allocate available neurons to the most informative parts of input documents.

To efficiently analyze the long-length patent documents, some studies have attempted to divide and explain it hierarchically. Hierarchical Feature Extraction Model (HFEM) [4] and Hierarchical Attention Networks (HAN) [26] have constructed hierarchical structures to extract document features sequentially from each word to a whole document. There are studies [27,28] using hierarchical patent classification information as a hybrid approach with deep learning and expert systems. These methods have the advantage of preventing inconsistencies in hierarchical classification results. However, since these methods uni-directional assign parent category results to child category results, it is difficult to utilize the hierarchical category’s correlation information in the multi-label classification of the APC task.

While those deep learning approaches on the patent documents have shown to be useful in embedding word features and classifying the long-length patent documents, they do not consider various- and long-length patent documents because of the limitation of computational cost. Patent documents are generally very lengthy documents. Since each paragraph has independent information of the patent claims and their description, every word and paragraph will likely have a powerful influence on the character of the patent documents. Therefore, analyzing the patent documents by receiving only a fixed-length input harms the performance of a network. To end this, we designed our model to be suitable for the patent document analysis by hierarchically extracting the whole document features and removing restrictions on the fixed-input length.

3. Multi-Stage Extraction Network: MEXN

Multi-stage feature extraction network (MEXN) is designed for extracting more informative documents features by adopting a hierarchical attention scheme. MEXN framework consists of paragraph encoder (P-Block), paragraphs summarizing encoder (D-Block), and label validation checker (L-Block).

Our framework takes a document as input and splits it into paragraphs \(P_1, \ldots, P_m\). Each of \(P_i\) converts all words into embedding vectors using pre-defined word embedding process [23]. \(P_i\) is then represented a fixed sized feature vector \(h_{P_i}\) in P-Block. After obtaining all \(h_P\), D-Block aggregates and summarizes them into 1-D vector by balancing between the paragraph features using hierarchy attention mechanism. The multi-stage process can handle input scalability by stacking feature extraction stages. By using this simple structure, we resolve to overcome the fixed-length input problem. Figure 1 and the following equations and represent the overall process:

\[
\begin{align*}
    h_{P_i} &= \mathcal{F}(P_i), \\
    v &= \mathcal{G}(H_P), \quad H_P \in \{h_{P_1}, \ldots, h_{P_m}\}, \\
    v' &= \mathcal{J}(v), \\
    p &= \text{Softmax}(W_v v' + b_v).
\end{align*}
\]
Equation (1) indicates feature encoding process for a paragraph (P-Block) which described in Section 3.2, Equation (2) presents the paragraph features aggregating process (D-Block) in Section 3.3. We also provide Equation (3) which enhances the classification accuracy using hierarchical label information (L-Block) in Section 3.4. Lastly, the aggregated features are then passed through a final fully connected layer and softmax to produce label predictions in Section 3.5.

3.1. Preprocessing

In our setting, we start by splitting a document into paragraphs before feeding it into MEXN. When using a self-attention method, the more input words are used, the better the performance will be [18]. However, it is not easy for a network to effectively handle such long-length inputs because the long-length inputs increase the computational cost and memory consumption. To alleviate this problem, we use a paragraph as a fundamental unit of feeding data.

Let \( \mathcal{T} = [T_1, T_2, \ldots, T_N] \) and \( \mathcal{L} = [L_1, L_2, \ldots, L_N] \) are the set of documents and their labels. Before feeding the data, we split each document into paragraphs with a fixed-size word length \( l \), and all words are embedded to \( D \) dimensional features. To prevent splitting the continuity of sentences, we designed one sentence not to be divided into two consecutive paragraphs. At the end of the paragraph, we fill in the unique tokens [PAD] with as many words as there are not enough words. For example, a paragraph with longer than length \( l \) will be splitted into two paragraphs. We also regard the title and summary parts as paragraphs. We denote the feature matrix for a paragraph as \( P_i \in \mathbb{R}^{D \times l} \) in Equation (1).

3.2. Paragraph Encoder (P-Block)

P-Block function \( \mathcal{F} \) encodes the feature matrix \( P_i \in \mathbb{R}^{D \times l} \) to a paragraph feature vector \( h_{P_i} \in \mathbb{R}^{D \times 1} \), which takes fixed-size length input which is pre-splitted paragraph by Preprocessing step. This operation summarizes the feature matrix \( P_i \) based on the composition of words and sentences only using its own contexts in the paragraph, which means \( \mathcal{F}(P_i) \) does not consider the information of neighbor paragraphs for the \( P_i \). The information of all other neighbor paragraphs will be aggregated in D-Block operation. P-Block can be easily implemented by CNN or RNN layers. Again, all words in paragraphs are embedded into the pre-defined feature vector with length \( l \). Next, we describe three different instances of P-Block operation, \( \mathcal{F} \).

CNN. A simple instance for P-Block is CNN, which is the straight-forward approach to apply to P-Block encoder because \( P_i \) is a fixed-size feature matrix:

\[
\mathcal{F}(P_i) = W_\theta P_i + b,
\]

where \( i \) is the index of paragraph position. \( W_\theta \) and \( b \) indicate learnable convolutional weight parameters and bias, respectively.
The traditional way to compute features from sequential inputs is RNN, which is not restricted in the input length size. It also can be applied for fixed size feature input.

\[ F(p_i) = \tanh(W_p h_p + W_p p_i + b), \]

where \( W_p, W_p \) and \( b \) indicate learnable RNN weight parameters and bias, respectively. Before feeding P-Block with CNN or RNN, we use Glove model [29] when embedding words in the paragraphs.

**Attention.** We can also design a stacked feature encoder such as Transformer mechanism [18], which extracts the paragraph feature hierarchically as:

\[ F(p_i) = \sum_{l=1}^{L} A_l p_i, \]

\[ [CLS]_i = A_l (Q_i^{l-1}, K_i^{l-1}, V_i^{l-1}), \]

where \( Q, K, \) and \( V \) indicate query, key, and value for the Transformer encoder, respectively. These feature vectors are also embedded by concatenating \([CLS]_i \) token and \( P_i \) as \( Q_i = K_i = V_i = \{[CLS], P_i\} \). Here, \{ ... \} denotes concatenation. \( L \) is the total number of stacked encoders, and \( l \) is the index of the stacked encoder. \( L \) and \( s \) determines how many encoder layers are used to summarizing paragraph \( P_i \). In attention based P-Block, unlike the previous two method (e.g., CNN, and RNN), we use contextualized word embedding method [30] as used by Bert [19].

### 3.3. Paragraphs Summarizing Encoder (D-Block)

D-Block \( G \) aims to summarize all paragraphs in a document and produce the document feature vector \( v \) in Equation (3) by adjusting balances between paragraph features \( h_p \). In general, the patent documents are composed of many paragraphs, thereby we adopt the attention mechanism which is specialized to help memorize long-length information and is a useful way to calibrate the weight of each feature’s contribution for classification. In this paper, we utilize the two most commonly used attention methods: additive attention [31] and productive attention approaches [18].

**Additive attention.** We use bi-directional GRU encoder [32] to exploit the sequence of paragraphs, which summarizes paragraph features \( h_p \) by the linear combination of the encoder states and the decoder states. Since this bi-directional GRU encoder is less susceptible to the gradient vanishing problem, it is suitable for processing long-length data like patent documents.

\[ G(h_p) = \sum_{i} a_i h_{D_i}, \quad a_i = \frac{\exp(u_i^\top u_s)}{\sum_{i}(\exp(u_i^\top u_s))}, \quad u_i = \tanh(W_s h_{D_i} + b_s), \]

where \( a_i \) is a weight of each paragraph feature \( h_{P_i} \), which is computed by dot product of the context vector \( u_i \) and state vector \( u_s \). The context vector \( u_i \) represents summarized paragraph feature, which can be derived from the hidden vector \( h_{D_i} \). This hidden vector is concatenated by bi-direction GRU encoder. The overall process is depicted in Figure 2a.

**Productive attention.** The stacked transformer encoder [18] is also used to summarize paragraph feature \( h_p \), which is an efficient way to summarize processing while computing cost is lower than additive attention. Unlike the additive attention, which processes the input features sequentially, this computes entire inputs at the one time with dot-product operation. This difference affects performance by changing the receptive field range that can be considered in one process.

\[ G(h_p) = A_{D_i} (Q_i^{l-1}, K_i^{l-1}, V_i^{l-1}), \]

where \( l \) is the number of stacked encoder \( A_{D_i} \). \( Q, K, \) and \( V \) indicate query, key, and value for the transformer encoder, respectively.
To aggregate paragraph features, symmetry operations (e.g., max, min, mean) and a classification token [DCLS] can be used. In the case of token, the feature vectors are embedded into the concatenated [DCLS] token. \( h_P \) can be computed as:

\[
Q_i = K_i = V_i = \{ [\text{DCLS}], h_P \}.
\]

By appending the token in front of the input paragraph features, it summarize paragraph features on each stacked layer.

### 3.4. Label Validation Checker (L-Block)

L-Block uses hierarchical label information of patents to enhance prediction reliability by comparing and verifying inter-dependency between the parent and child categories. To boost the probability of prediction, we reflect the predictions for the parent category to the child category predictions. We design L-Block as follows so that the label dependency relation can be considered:

\[
v' = \mathcal{F}(v) = \text{Broadcast}(v_p) \times v_c,
\]

where \( v_p \in \mathbb{R}^{N_p} \) and \( v_c \in \mathbb{R}^{N_c} \) are the predicted weight vectors of the parent and child categories. Because the number of the two categories are different, we expand the vector dimension of the parent by the size of the child category, and broadcast the vector weights of the parent to the expanded vector. The weight vectors of the parent and child are obtained by:

\[
v_p = W_p v + b_p \quad \text{and} \quad v_c = W_c v + b_c,
\]

where \( W \) and \( b \) are weight matrices and biases to be learned.

### 3.5. Document Classification

The end of the network is the single fully connected (FC) network for document classification. The output of the last FC is fed into a 9-way and a 128-way softmax, which produces distribution over the 9 parent and 128 child categories, respectively. MEXN is optimized by BCEWithLogitsLoss for multi-label and multi-class classification problem as follows:

\[
l(x, y) = L = \{ l_1, \ldots, l_N \}, \quad l_n = -\left[ y_n \cdot \log \sigma(x_n) + (1 - y_n) \cdot \log(1 - \sigma(x_n)) \right],
\]

where target \( y \), predicted label \( x \) and sigmoid function \( \sigma \) for coordinate training loss.

### 4. Experimental Results

For performance evaluation, we report the performance of MEXN comparing with recent learning-based methods. For fair comparisons of multi-label classification problems, we adopt three different evaluation metrics: matching score, hamming loss, and Jaccard similarity.
4.1. Dataset

We evaluate MEXN on two different datasets: USPD and AAPD. Table 1 shows the simple statistical information of the datasets. C denotes the number of classes in the dataset, N the number of samples, and W and S the average number of words and sentences per document, respectively.

**USPD.** ([http://www.patentsview.org/download/](http://www.patentsview.org/download/)) We use the U.S. Patented Dataset (USPD) from 1976 to 2020 provided by the United States Patent and Trademark Office (USPTO) as a data source, which includes 33,931,998 classified patents and 91,920,308 claims. USPD categorizes the patent documents into hierarchical levels (9 sections, 128 sub-sections, 656 main-groups, and 259,048 sub-groups). Each document can belong in multiple categories at each level.

Each patent document consists of a unique patent number, a title, a summary, multiple claims, and descriptions for claims. For fair comparative evaluation with other methods, we exclude images in the patent documents, then evaluate classification performance for section and sub-section. In the main/sub-group categories, the total number of labels for each category is not enough amount and uniform to train. So they were not used in our evaluation.

**AAPD.** ([https://github.com/lancopku/SGM](https://github.com/lancopku/SGM)) We also use Arxiv Academic Paper Dataset (AAPD), which contains the abstract of 55,840 papers in the computer science field. The academic papers are categorized by multiple corresponding subjects. There are 54 subjects in total, and 163.42 words and 2.41 labels per each data sample on average.

| Dataset | C | N    | W    | S    |
|---------|---|------|------|------|
| AAPD    | 54| 55,840| 163.42| 1.0  |
| USPD    |   |       |      |      |
| - section | 9 | 351,009| 1497.2| 56.9 |
| - subsection | 128 | 351,009| 1497.2| 56.9 |

4.2. Metrics

**Exact matching score (EM)** denotes the percentage of correctly classified samples. This measurement partially shows multi-label classification performance, but also partially ignore correct matches. Therefore, we use the following two methods as additional criteria for comparing performance evaluation.

\[
EM = \frac{1}{N} \sum_{i=1}^{N} I(Y_i = Z_i), \quad I = \begin{cases} 
1 & \text{if } Y = Z, \\
0 & \text{if } Y \neq Z
\end{cases},
\]

where \(Z_i\) and \(Y_i\) denotes the \(i\)-th target and prediction, respectively.

**Hamming loss (HL)** is the fraction of labels that are incorrectly predicted [34] (i.e., the ratio of the wrong predicted labels to the total number of labels).

\[
HL = \frac{1}{NL} \sum_{i=1}^{L} \sum_{j=1}^{N} Y_{ij} \oplus Z_{ij},
\]

where \(\oplus\) indicates XOR operation, \(Z_{ij}\) and \(Y_{ij}\) denotes the \(i\)-th target and prediction respectively, which contains the \(l\)-th label. The lower HL, the better performance.
**Jaccard similarity as an accuracy (Jacc)** measures similarities between the target label sets and the predicted sets, which is the useful performance metric to evaluate multi-label classification problem [35].

\[
Jacc = \frac{1}{N} \sum_{i=1}^{N} \frac{|Y_i \cap Z_i|}{|Y_i \cup Z_i|}.
\]  

(15)

### 4.3. Implementation Details

We train networks for 100 epochs with an initial learning rate is set to 0.001 and divided by 0.1 every 30 epoch. USPD is consist of 334,705 training and 16,304 validation documents. In our experiments, we use 768 dimensions of embedding vector size \(D\) for each word, paragraph feature, and set the input size \(l\) to 500 words, which is the maximum input size of Bert [19]. We evaluate MEXN for 10 times with difference random seeds, then report the average performances. We use Adam optimizer [36] and apply early stopping method by EM score. We used a 3.70-GHz hexa-core CPU, 64 GB RAM, and RTX-2080Ti 12GB GPU RAM for the implementation.

### 4.4. Ablation Studies

#### 4.4.1. Ablation Study on P-Block

**P-Block variations.** We investigate on various of P-Block to evaluate the performance of feature extraction for paragraphs. It can be seen in Table 2 that for four difference instances of P-Blocks (e.g., CNN model [1], RNN(bi-directional LSTM) model [25], pre-trained, and fine-tuned Bert model [19]). The pre-trained Bert model is learned by the Bookcorpus [37] (800 M words) and English Wikipedia (2500 M words) The fine-tuned Bert is trained by USPD. Since the performance of the pre-trained model shows superior performance over other instances, we adopt the pre-trained model as P-Block in the following experiments. Note that even if using the pre-trained Bert, we can see that it also shows the promised performance, which means our MEXN has the capability of dealing with documents in a specific domain (e.g., patents) by using generalized word embeddings.

|                | Section | Subsection |
|----------------|---------|------------|
|                | EM      | HL         | Jacc     | EM      | HL         | Jacc     |
| CNN            | 49.981  | 0.0788     | 68.424   | 26.395  | 0.00984    | 35.943   |
| RNN(bi-LSTM)   | 56.793  | 0.0745     | 71.594   | 36.531  | 0.00843    | 53.235   |
| pre-trained BERT| 54.954  | 0.0697     | 73.696   | 39.861  | 0.00836    | 58.364   |
| fine-tuned BERT| 66.692  | 0.0471     | 82.710   | 51.565  | 0.00616    | 72.716   |

#### 4.4.2. Ablation Study on D-Block

**D-Block variations.** We experiment on variations of D-Block to demonstrate the performance of summarizing paragraphs information. To fairly evaluate the addictive and productive attentions, we use the fine-tuned Bert for P-Block as described in the previous section, and L-Block that cross-validates the labels of section and sub-section. In Table 3, we observe that the productive method shows better performance compared to the addictive attention. Productive attention exploits more directly computing attention weight between paragraphs than in additive attention, which sequentially propagates information via RNN units so that productive attention can take into account the contextual significance of each paragraph well.
Summarizing methods. We also study the classification token method by comparing it with feature pooling approaches in D-Block in Table 3. The productive attention uses a classification token, $[DCLS]$, to summarize a document vector, and other compared methods aggregate paragraph features by pooling. We observe that the classification token shows better classification performance than others. The results show that the token method is also a useful approach with the transformer encoder process for summarizing feature vectors like $[CLS]$ token on word embedding classification.

4.4.3. Ablation Study on L-Block

Label validator. To evaluate L-Block stage model, we perform sub-section category classification with L-Block process. The experiment shows the task effect of applying a parent-child relationship with the label hierarchy information. We use section and sub-section hierarchy labels for subsection label classification. In Table 4, we observe that the EM score and HL score is within a margin of error, but the model with L-Block is better performed on Jacc than without L-Block.

4.5. Parametric Study

Limited length of document. To evaluate limited input size for classification, we present classification results on the AAPD dataset. We hire our model with productive attention on D-Block. In Table 5, we observe that our model has a similar performance within the margin of error with patentBert. It is because MEXN and patentBert proceed through the same method that uses a single paragraph feature.

The number of paragraphs. To evaluate performance as the number of input paragraphs changes, we train our model with various input paragraph size. Considering the average size of the USPD dataset and limitation of memory performance, we set the maximum number of paragraphs at eight. Our method trains by increasing the number of paragraphs from 2 to 8 in the section and sub-section...
category each. In Figure 3, we observe that our model shows the best performance when using the maximum number of paragraphs on the USPD dataset.

![Figure 3. Average Jacc according to the number of paragraphs.](image)

**The paragraph length.** As described in Section 3.1, for computational and memory efficiency, we split the paragraph into constant size during the preprocessing step and use it as input data. In this section, we analyze the performance of the proposed MEXN according to the length of the split input paragraph. In this experiment, we set the length of the input paragraph with three different lengths: 100, 250, and 500. Our model uses the attention operation [19] in the P-Block for optimal performance. In general, this attention operation positively has a positive effect on classification performance as the input length increases [18], and as shown in Table 6, our model setting shows similar trends. However, given the limitations of computing memory, the increase in the input is a problem that computational cost increases multiply due to productive operation. Hence, we can choose the paragraph size depending on the system performance. We have chosen the best paragraph size that our system allows.

**Selecting part of patent document.** We also evaluate the performance according to the combination of input paragraphs (e.g., title, abstract, and main context). Table 7 shows the best performance when using the whole document. The result reveals that it is necessary to consider various parts of the document together to better understand the long-length documents.

| Section | Subsection | Memory Usage per 1 Batch (MB) |
|---------|------------|-------------------------------|
| EM      | HL Jacc    | EM   | HL Jacc    | EM   | HL Jacc |
| paragraph:100 | 62.384 | 0.0546 | 79.852 | 47.655 | 0.0070 | 68.766 | 120 |
| 250 | 64.457 | 0.0511 | 81.167 | 49.483 | 0.0069 | 70.329 | 258 |
| full text | 66.692 | 0.0471 | 82.710 | 51.565 | 0.0062 | 72.716 | 809 |

| Title | Jacc | Title + abstract | Jacc | Full text | Jacc |
|-------|------|------------------|------|-----------|------|
| EM    | HL   | EM   | HL | Jacc | EM   | HL | Jacc |
| title | 62.349 | 0.0532 | 80.328 | 48.497 | 0.0068 | 69.677 |
| title + abstract | 65.541 | 0.0488 | 82.061 | 50.510 | 0.0064 | 71.835 |
| full text | 66.692 | 0.0471 | 82.710 | 51.565 | 0.0062 | 72.716 |
4.6. Comparisons with State-of-the-Arts on USPD

We compare MEXN to other document classification methods on the USPD validation set. Table 8 shows the classification performances on three different categories (i.e., section, subsection, and overall). Overall is the union of section and subsection results. We can see that MEXN outperforms other methods in all evaluation metrics by large margins. In this experiment, MEXN adopts fine-tuned Bert for P-Block and productive attention for D-Block, which shows the best performance on ablation studies. Note that even if we do not apply L-Block to our model, MEXN shows the improving classification results.

Table 8. Comparisons on USPD validation set.

|        | Section | Subsection | Overall |
|--------|---------|------------|---------|
|        | EM      | HL         | Jacc    | EM      | HL         | Jacc    | EM      | HL         | Jacc    |
| kim-CNN| 48.613  | 0.0813     | 66.221  | 24.899  | 0.0111    | 34.742  | 20.058  | 0.0157    | 48.500  |
| regLSTM| 53.756  | 0.0716     | 71.5755 | 35.821  | 0.0093    | 0.042   | 31.021  | 0.0133    | 58.799  |
| HAN    | 51.441  | 0.0760     | 71.462  | 37.855  | 0.0090    | 56.976  | 30.863  | 0.0134    | 61.778  |
| patentBert | 64.891 | 0.0502     | 81.482  | 49.976  | 0.0066    | 70.833  | 45.435  | 0.0094    | 74.329  |
| MEXN   | 66.692  | 0.0471     | 82.710  | 51.565  | 0.0062    | 72.716  | 45.813  | 0.0090    | 75.489  |

4.7. Computational Cost

We analyze the computational cost of MEXN by comparison with patentBert. The computational cost of patentBert is 37.82 G $\text{FLOPs}$ for a single paragraph, which theoretically increases by multiplication when computing two or more paragraph inputs. However, MEXN has a marginal increase in the computational cost in run-time, even if the number of paragraphs increases due to its hierarchical and parallel feature extraction structure. Figure 4 shows the comparison of the total amount and run-time of computational cost with MEXN and patentBert. MEXN takes 38.16 G $\text{FLOPs}$ of the computational cost on the maximum number of paragraphs. Furthermore, it is only 0.006 percent increase compared with patentBert on a single paragraph input size.

![Computational cost](image)

Figure 4. Computational cost analysis.

4.8. Visualization

**Attention weight curve.** We experiment to investigate the distribution of the attention weight for a deeper understanding on MEXN. When the distribution of the weights is uniform, the classification
outputs are equally support by all paragraphs. On the other hand, when the attention weight
distribution concentrates in one paragraph, MEXN performs the classification by focusing on that
paragraph. We calculate the weight values for each paragraph of the document in the validation set
and then average them to produce distribution. In Figure 5, we observe the skewed distribution of
weight at the section categories. It means that the model set more weight title and abstract parts of
the document. On the other hand, the sub-section category shows a more uniform distribution than
the section category. This indicates that the detailed descriptions are more helpful to understand the
documents more specifically.

![Attention weight Distribution](image)

**Figure 5.** Paragraph Attention Distribution.

5. Conclusions

In this work, we presented Multi-stage Extraction Network (MEXN) to classify long-length
patent documents, which enables input without limit. The proposed MEXN consists of paragraph
coder (P-Block), paragraph summarizing encoder (D-Block), and label validation checker (L-Block)
based on the hierarchical structure, which breaks the limit of cropped input document on the deep
learning classification task. We provided ablation studies for a better understanding of MEXN and
experiments on large benchmark datasets to demonstrate the performance of MEXN.

We are excited about the future of our model to be applied to other formal document classification
tasks. We also plan to adopt multi-modal inputs (e.g., images, sounds, and video) to analyze various
multi-media data.

**Author Contributions:** Conceptualization, J.B.; methodology, J.B.; software, J.B.; validation, J.B. and I.S.; formal
analysis, J.B.; investigation, J.B.; resources, J.B.; data curation, J.B.; writing—original draft preparation, J.B. and I.S.;
writing—review and editing, J.B. and I.S.; visualization, J.B.; supervision, I.S. and S.P.; project administration, I.S.
and S.P.; funding acquisition, S.P. All authors have read and agreed to the published version of the manuscript.

**Funding:** This work was supported by the National Research Foundation of Korea (NRF) grant funded by the
Korea government (MIST) (No. NRF-2017R1D1A1B03036252).

**Conflicts of Interest:** The authors declare no conflict of interest.
References

1. Kim, Y. Convolutional neural networks for sentence classification. *arXiv* 2014, arXiv:1408.5882.
2. Li, S.; Hu, J.; Cui, Y.; Hu, J. DeepPatent: Patent classification with convolutional neural networks and word embedding. *Scientometrics* 2018, 117, 721–744. [CrossRef]
3. Lee, J.S.; Hsiang, J. PatentBERT: Patent Classification with Fine-Tuning a pre-trained BERT Model. *arXiv* 2019, arXiv:1906.02124.
4. Hu, J.; Li, S.; Hu, J.; Yang, G. A Hierarchical Feature Extraction Model for Multi-Label Mechanical Patent Classification. *Sustainability* 2018, 10, 219. [CrossRef]
5. Xiao, L.; Wang, G.; Zuo, Y. Research on Patent Text Classification Based on Word2Vec and LSTM. In Proceedings of the 2018 11th International Symposium on Computational Intelligence and Design (ISCID), Hangzhou, China, 8–9 December 2018; Volume 1, pp. 71–74.
6. Ma, X.; Hovy, E. End-to-end sequence labeling via bi-directional lstm-cnns-crf. *arXiv* 2016, arXiv:1603.01354.
7. Hochreiter, S.; Schmidhuber, J. Long short-term memory. *Neural Comput.* 1997, 9, 1735–1780. [CrossRef]
8. Jackson, P. *Introduction to Expert Systems*; Addison-Wesley Longman Publishing Co., Inc.: Boston, MA, USA, 1998.
9. Kim, J.H.; Choi, K.S. Patent document categorization based on semantic structural information. *Inf. Process. Manag.* 2007, 43, 1200–1215. [CrossRef]
10. Roh, T.; Jeong, Y.; Yoon, B. Developing a Methodology of Structuring and Layering Technological Information in Patent Documents through Natural Language Processing. *Sustainability* 2017, 9, 2117. [CrossRef]
11. Lim, J.; Choi, S.; Lim, C.; Kim, K. SAO-based semantic mining of patents for semi-automatic construction of a customer job map. *Sustainability* 2017, 9, 1386. [CrossRef]
12. Fall, C.J.; Töröcskévi, A.; Benzineb, K.; Karetka, G. Automated categorization in the international patent classification. In *Acm Sigir Forum*; ACM: New York, NY, USA, 2003; Volume 37, pp. 10–25.
13. Wu, C.H.; Ken, Y.; Huang, T. Patent classification system using a new hybrid genetic algorithm support vector machine. *Appl. Soft Comput.* 2010, 10, 1164–1177. [CrossRef]
14. Kim, G.; Lee, J.; Jang, D.; Park, S. Technology clusters exploration for patent portfolio through patent abstract analysis. *Sustainability* 2016, 8, 1252. [CrossRef]
15. Robertson, S. Understanding inverse document frequency: On theoretical arguments for IDF. *J. Doc.* 2004, 60, 503–520. [CrossRef]
16. Cortes, C.; Vapnik, V. Support-vector networks. *Mach. Learn.* 1995, 20, 273–297. [CrossRef]
17. Minnaert, S.; Kalchbrenner, N.; Cambria, E.; Nikzad, N.; Chenaghlu, M.; Gao, J. Deep learning based text classification: A comprehensive review. *arXiv* 2020, arXiv:2004.03705.
18. Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A.N.; Kaiser, L.; Polosukhin, I. Attention is all you need. In *Advances in Neural Information Processing Systems*; MIT Press: Cambridge, MA, USA, 2017; pp. 5998–6008.
19. Devlin, J.; Chang, M.W.; Lee, K.; Toutanova, K. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv* 2018, arXiv:1810.04805.
20. Dai, Z.; Yang, Z.; Yang, Y.; Carbonell, J.G.; Le, Q.; Salakhutdinov, R. Transformer-XL: Attentive Language Models beyond a Fixed-Length Context. In Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, Florence, Italy, 28 July–2 August 2019; pp. 2978–2988.
21. Yang, Z.; Dai, Z.; Yang, Y.; Carbonell, J.; Salakhutdinov, R.R.; Le, Q.V. Xlnet: Generalized autoregressive pretraining for language understanding. In *Advances in Neural Information Processing Systems*; MIT Press: Cambridge, MA, USA, 2019; pp. 5753–5763.
22. Beltagy, I.; Peters, M.E.; Cohan, A. Longformer: The long-document transformer. *arXiv* 2020, arXiv:2004.05150.
23. Mikolov, T.; Chen, K.; Corrado, G.; Dean, J. Efficient estimation of word representations in vector space. *arXiv* 2013, arXiv:1301.3781.
24. Risch, J.; Krestel, R. Domain-specific word embeddings for patent classification. *Data Technol. Appl.* 2019, 53, 108–122. [CrossRef]
25. Schuster, M.; Paliwal, K.K. Bidirectional recurrent neural networks. *IEEE Trans. Signal Process.* 1997, 45, 2673–2681. [CrossRef]
26. Yang, Z.; Yang, D.; Dyer, C.; He, X.; Smola, A.; Hovy, E. Hierarchical attention networks for document classification. In Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, San Diego, CA, USA, 12–17 June 2016; pp. 1480–1489.

27. Roudsari, A.H.; Afshar, J.; Lee, C.C.; Lee, W. Multi-label Patent Classification using Attention-Aware Deep Learning Model. In Proceedings of the IEEE International Conference on Big Data and Smart Computing (BigComp), Busan, Korea, 19–22 February 2020; pp. 558–559.

28. Risch, J.; Garda, S.; Krestel, R. Hierarchical Document Classification as a Sequence Generation Task. In Proceedings of the ACM/IEEE Joint Conference on Digital Libraries, Wuhan, China, 19–23 June 2020; pp. 147–155.

29. Pennington, J.; Socher, R.; Manning, C. Glove: Global vectors for word representation. In Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP), Doha, Qatar, 25–29 October 2014; pp. 1532–1543.

30. Young, T.; Hazarika, D.; Poria, S.; Cambria, E. Recent trends in deep learning based natural language processing. *IEEE Comput. Intell. Mag.* 2018, 13, 55–75. [CrossRef]

31. Bahdanau, D.; Cho, K.; Bengio, Y. Neural machine translation by jointly learning to align and translate. *arXiv* 2014, arXiv:1409.0473.

32. Chung, J.; Gulcehre, C.; Cho, K.; Bengio, Y. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv* 2014, arXiv:1412.3555.

33. Yang, P.; Sun, X.; Li, W.; Ma, S.; Wu, W.; Wang, H. SGM: Sequence generation model for multi-label classification. *arXiv* 2018, arXiv:1806.04822.

34. Schapire, R.E.; Singer, Y. Improved boosting algorithms using confidence-rated predictions. *Mach. Learn.* 1999, 37, 297–336. [CrossRef]

35. Boutell, M.R.; Luo, J.; Shen, X.; Brown, C.M. Learning multi-label scene classification. *Pattern Recognit.* 2004, 37, 1757–1771. [CrossRef]

36. Kingma, D.P.; Ba, J. Adam: A method for stochastic optimization. *arXiv* 2014, arXiv:1412.6980.

37. Zhu, Y.; Kiros, R.; Zemel, R.; Salakhutdinov, R.; Urtasun, R.; Torralba, A.; Fidler, S. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In Proceedings of the IEEE International Conference on Computer Vision, Santiago, Chile, 7–13 December 2015; pp. 19–27.

© 2020 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (http://creativecommons.org/licenses/by/4.0/).