PatentNet: multi-label classification of patent documents using deep learning based language understanding

Arousha Haghighian Roudsari¹ · Jafar Afshar¹ · Wookey Lee^{1,2} · Suan Lee³

Received: 12 November 2020 / Accepted: 7 October 2021 / Published online: 18 December 2021
© The Author(s) 2021

Abstract

Patent classification is an expensive and time-consuming task that has conventionally been performed by domain experts. However, the increase in the number of filed patents and the complexity of the documents make the classification task challenging. The text used in patent documents is not always written in a way to efficiently convey knowledge. Moreover, patent classification is a multi-label classification task with a large number of labels, which makes the problem even more complicated. Hence, automating this expensive and laborious task is essential for assisting domain experts in managing patent documents, facilitating reliable search, retrieval, and further patent analysis tasks. Transfer learning and pre-trained language models have recently achieved state-of-the-art results in many Natural Language Processing tasks. In this work, we focus on investigating the effect of fine-tuning the pre-trained language models, namely, BERT, XLNet, RoBERTa, and ELECTRA, for the essential task of multi-label patent classification. We compare these models with the baseline deep-learning approaches used for patent classification. We use various word embeddings to enhance the performance of the baseline models. The publicly available USPTO-2M patent classification benchmark and M-patent datasets are used for conducting experiments. We conclude that fine-tuning the pre-trained language models on the patent text improves the multi-label patent classification performance. Our findings indicate that XLNet performs the best and achieves a new state-of-the-art classification performance with respect to precision, recall, F1 measure, as well as coverage error, and LRAP.

Keywords Patent classification · Multi-label text classification · Pre-trained language model

✉ Wookey Lee
trinity@inha.ac.kr

Arousha Haghighian Roudsari
arousha.haghighian@inha.edu

¹ Department of Industrial Engineering, Inha University, Incheon, South Korea

² Department of Biomedical Science and Engineering, Inha University, Incheon, South Korea

³ School of Computer Science, Semyung University, Jecheon, South Korea

Introduction

Patent documents contain valuable information and, if scrutinized, can reveal substantial technical details, inspire new industrial solutions, depict leading business trends, and assist in making critical investment decisions (Zhang et al., 2015). Patent analysis tasks, be it technology exploration, prior art search, or preventing possible patent infringement, require domain-specific knowledge and has long been performed by domain experts and patent attorneys. However, the rapid growth and development in different technology areas have led to a significant increase in patent applications in recent years. This increase in the number of patent documents makes patent analysis and management more complicated and time-consuming for patent experts and examiners, posing significant challenges for many patent information users (Yun & Geum, 2020; Chen et al., 2020b). Thus, automated technologies for assisting patent experts and patent information users in patent analysis related tasks are in high demand.

One of the primary steps for managing patents is classification, where patents about similar topics or similar technological areas are classified under the same category. Common standard classification structures such as the International Patent Classification (IPC) or Cooperative Patent Classification (CPC) are used for patent classification (Shalaby & Zadrozny, 2019). These standard taxonomies consist of complex hierarchical structures that cover all technology areas and help maintain inter-operability among various patent offices worldwide (Gomez & Moens, 2014). Therefore, accurate automated classification of patent documents is critical and will help experts manage patent documents, facilitate reliable patent search and retrieval, reduce the risk of missing a relevant patent in preventing patent infringement, and further patent analysis tasks (Yun & Geum, 2020; Souza et al., 2020; Gomez & Moens, 2014).

However, utilizing traditional text processing methods has not successfully processed patent text and effectively extracted features from it for patent search, retrieval, and classification (Shalaby et al., 2018). This is because patents are lengthy and complicated legal documents that usually follow a fixed structure and include multiple parts such as meta-data, title, abstract, description, claims, representative drawings, and citations for describing the novelty or inventive steps of the invention. Nevertheless, unlike other academic or scientific publications, the patents' text might not be written in a way to convey knowledge explicitly. The deliberate usage of vague and new terms in patent documents are intended to protect and keep as broad as possible rights for the contained intellectual property (D'hondt et al., 2017; Shalaby et al., 2018; Gomez & Moens, 2014). The patent document is full of jargon, complex, or new technical words. Moreover, multi-word units or phrasal expressions are used to refer to the same concepts. It can be a combination of general terms or functional terms instead of using a single word representation (D'hondt et al., 2013). For example, "electric current carrier" is substituted for the word "wire". Besides, the same word may have different meanings in different technology areas. For example, the word "property" implies something that belongs to someone in legislation, while meaning as characteristics of something in chemistry field (Chen et al., 2020a).

Moreover, as an invention may pertain to multiple technology areas, a patent can be assigned to several classification labels. Therefore, patent classification becomes a multi-label classification problem which is more challenging. The reasons are threefold. First, the IPC consists of a complex hierarchical structure with a large number of labels. For example, at the IPC subclass level, the number of labels is 645, which further increases

to around 67,000 at the sub-group level¹. Second, the distribution of patents in these categories is imbalanced. As knowledge and technologies tend to change over time, patent classification structures also change thereon. New categories may appear in the classification structures, or the existing categories that have experienced little use are combined with other categories (Lupu & Hanbury, 2013). Third, multi-label classification is different and harder to learn than the binary and multi-class classification problems. The document representations should capture much richer information to correctly predict multiple correlated labels and distinguish them from large numbers of irrelevant labels (Liu et al., 2017).

Previous attempts for automating patent classification have been based on traditional text or document representation methods (such as Bag-of-Words, TF-IDF, and N-grams) and machine learning classification models (such as SVM, K-Nearest Neighbor, and Naive Bayes) (Fall et al., 2003; Wu et al., 2010; D'hondt et al., 2013), all with limited success that is achieved through tedious hand-crafted feature extractions for representing the patent document. However, features can be extracted automatically by deep learning models. Recently, some studies have implemented deep learning models such as convolutional neural network (CNN) (Li et al., 2018; Abdelgawad et al., 2019), recurrent neural network (RNN) based models (Grawe et al., 2017; Risch & Krestel, 2019), and their combination Hu et al. (2018a) for automated patent classification. These models convert the input text to a feature vector representation considering the context by extracting higher-level features from word vectors (Minaee et al., 2020).

Despite the progress, the proposed deep learning models for patent classification still have some limitations. Even though these models have been successful in extracting features for various Natural Language Processing (NLP) tasks, such as sentiment analysis, machine translation, and text classification, their performances are not very satisfactory when applied to patent data. This is due to the unique characteristics of the patent documents. These models are not powerful enough to encode the dependencies in the patent text and understand the multiple word units used to refer to the same meaning or concepts. Besides, the word embeddings (e.g., word2vec (Mikolov et al., 2013)), which are the basis of the proposed deep-learning models, may face the out-of-vocabulary problem and cannot account for polysemy. These issues are often faced when analyzing the patent text due to the specific language used in these documents. Therefore, it is crucial to use models that can distinguish between different meanings of the same word and are more powerful in language understanding to encode a rich contextual representation of patent text.

Recently, Transformer architecture proposed by Vaswani et al. (2017), which is based only on the attention mechanism, has made it possible to train large models with big data on GPUs efficiently. The Transformer avoids the recurrence and convolutions entirely and allows for much more parallelized computations than RNNs and CNNs. This has led to the appearance of Transformer-based pre-trained language models such as BERT (Devlin et al., 2018), XLNet (Yang et al., 2019), RoBERTa (Liu et al., 2019), ELECTRA (Clark et al., 2020), OpenAI GPT-3 Brown et al. (2020), among many others. These language models are composed of deep model architectures pre-trained on large text corpus, which can be fine-tuned on downstream tasks and have led to state-of-the-art results in many NLP tasks. However, the effectiveness of these pre-trained language models on patent documents that have unique characteristics is still not clear and needs to be investigated.

¹ <https://www.wipo.int/classifications/ipc/en/ITsupport/Version20190101/transformations/stats.html>.

Furthermore, it is hard to compare the existing automated patent classification methods due to the variations in problem definitions, dataset, IPC levels and patent sections considered for the classification task. Some studies limit the multi-label problem to binary or multi-class problems, which contrasts with the patent classification problem's characteristics. A consistent benchmark dataset is not used throughout the literature, and the size of the datasets vary widely. Moreover, previous studies have also failed to present various evaluation measures suitable for the performance of the multi-label classification problem. Thus, in this paper, we aim to improve automated patent classification by addressing the limitations of previous studies and using powerful language understanding models. We propose to utilize the latest pre-trained language models and fine-tune them for the multi-label patent classification task.

Our main contributions are as follows: (1) We fine-tuned and compared the pre-trained language models, namely, BERT, XLNet, RoBERTa, and ELECTRA, for the multi-label patent classification problem without simplifying the problem to binary or multi-class problems. (2) We demonstrated the superiority of the pre-trained language models by comparing them with other deep learning-based models proposed for the patent classification problem. (3) For the previous deep learning models, we further investigated the effect of using various word embeddings to improve the classification performance. (4) We evaluated the experimental results using several measures that are better suited for the multi-label classification task. Experimental results on two patent datasets indicate that XLNet, having the ability to model bi-directional context, performs the best on patent text and achieves a new state-of-the-art result on patent classification. (5) By comparing and analyzing various models and techniques, this paper's findings can be used as a guideline for other researchers for improving other patent analysis-related tasks.

The rest of the paper is composed as follows. In Sect. [Related works](#), some related work in automatic patent classification is provided. Section [Pre-trained language models](#) describes the various pre-trained language models. The multi-label patent classification problem and the fine-tuning process are explained in Sect. [Multi-label patent classification](#). Following this, in Sect. [Experiment](#), we provide the dataset, evaluation measures, baseline models, experimental setup, and the results. Finally, we conclude the paper in Section [Conclusion](#).

Related works

Previous studies have addressed the problem of patent classification from different perspectives. Some focused on the best way to represent the patent text and how to extract semantic features from it (D'hondt et al., 2013; Shalaby et al., 2018; Hu et al., 2018a; Hu et al., 2018b; Li et al., 2018) while others focused on designing more effective classification algorithms (Fall et al., 2003; Al Shamsi & Aung, 2016; D'hondt et al., 2017; Wu et al., 2010, 2016; Song et al., 2019). Furthermore, some attempts have been made to find which part of the patent text can be more representative and provide better classification results (Gomez, 2019; Hu et al., 2018a; Wu et al., 2010; D'hondt & Verberne, 2010). Gomez & Moens (2014) did a comprehensive survey of several previous works that tackled the automated patent classification problem in the IPC hierarchy. They pointed out the task's challenges and listed the previous works that mainly utilized traditional and classical methods. However, we will focus more on recent deep learning-based approaches for automated patent classification.

The most common approach to represent a text document was Bag of Words (BOW) methods, which were based on the term frequency vector of the available words in the documents. Term Frequency-Inverse Document Frequency (TF-IDF) is the most famous BOW measure. However, they neglect the semantics of the word, and similar documents will not be considered similar if the terms used in them do not overlap enough (Shalaby et al., 2018). To overcome the shortcomings of term frequency features, Mikolov et al. (2013) proposed the word2vec model for learning distributed representations of words in low dimensional spaces. It provides an embedding for each word that captures both the semantic and syntactic relationships between the words. A similar approach called, GloVe (Pennington et al., 2014), provides word embedding based on the global word-word co-occurrence counts rather than the fixed-sized windows of context. The limitations of these word embeddings are the out-of-vocabulary problem, i.e., if a new word appears in the test data that was not seen before in the training data, it will not have a vector representation in the word embedding space. Moreover, they cannot account for polysemy, and each word, regardless of having multiple meanings, is assigned with only one distinct vector representation. Bojanowski et al. (2017) proposed another word embedding approach, called fastText, to overcome the out of vocabulary problem. The fastText incorporates character-level information of words in the learning process.

Nevertheless, these embedding vectors, combined with other neural networks such as RNN and CNNs, have successfully achieved good results on various NLP tasks such as text classification (Minaee et al., 2020). The word embeddings are the basis of the deep learning models. The deep learning models extract higher-level features from these word embeddings for the downstream NLP task. The available pre-trained word embeddings, such as Google's word2vec trained on the large Google News dataset, have provided significant improvements over embeddings learned from scratch in many NLP tasks. However, the patent text is different and full of jargon, complex, and new technical terms. Therefore, in the patent domain, researchers use the embedding methods to train word embeddings using the patent corpus for obtaining better word representations. CNN can extract important n-gram features from the input sentence and, therefore, produce an information latent semantic representation of the input sequences for the downstream task. Meanwhile, RNNs are well known for performing sequential processing of the text by modeling units of the sequence and are able to capture the sequential nature that lies in a language (Young et al., 2018). More efficient forms of RNN, are namely, Gated Recurrent Units (GRU) and Long Short Term Memory (LSTM) networks.

Risch and Krestel (2019) trained a word embedding representation using the fastText approach on more than 5 million patent documents to obtain domain-specific embeddings for words in the patent domain. They then fed the embeddings to a bidirectional GRU architecture for the patent classification task. However, they only considered a multi-class classification task by considering one main subclass label of the IPC taxonomy for each patent document. Shalaby et al. (2018) proposed a novel classification approach that represents patent documents and their structure as a Fixed Hierarchy Vectors (FHV) model and then used a single-layered LSTM architecture for multi-label classification. They conducted experiments on three levels of the IPC classification hierarchy, namely, section, class, and subclass. Hu et al. (2018a) used word2vec word embedding and proposed a hybrid model for multi-label patent classification but only considering mechanical patents. Their hybrid model combines CNN and BiLSTM model for hierarchical feature extraction. Grawe et al. (2017) also used word2vec and LSTM for classifying patents but only used a small dataset. Li et al. (2018) proposed a deep learning algorithm, DeepPatent, using the combination of word embedding (word2vec) and the famous convolutional neural networks

(CNN) for the patent classification task. DeepPatent outperformed other algorithms from the CLEF-IP (Piroi et al., 2011) competition and contributed the new USPTO-2M dataset, which is much larger than the previous benchmarks. They tested different sections of patent documents and also the number of words that led to a better classification result. They concluded that only the first 100 words of both title and abstract can achieve good performance. Moreover, Abdelgawad et al. (2019) applied some recent deep learning models and investigated the effect of different word embeddings and neural network optimization on the patent classification task. However, they limited their scope of the study to a single-label classification task. Multi-label classification is harder than binary and multi-class classification problems. Much richer information should be captured in the document representation to correctly predict various correlated labels and differentiate them from large numbers of irrelevant labels. Roudsari et al. (2021) used a CNN model and investigated the effect of utilizing static versus contextual word embeddings on multi-label patent classification but only considered 89 labels at the IPC subclass level. Lee and Hsiang (2019) adopted the fine-tuning approach and used the pre-trained BERT model for the patent classification task. However, the scope of the study is limited, and they focused more on the CPC subclass classification and patent claim section due to their future work, which was the downstream task of patent claim generation. Therefore, a more extensive study needs to be conducted to compare and examine the effectiveness of pre-trained language models on the task of patent classification.

The appearance of pre-trained language models and transfer learning has significantly boosted text classification performance in NLP. However, the patent text is unique, filled with technical, vague, and new terms, which makes the multi-label classification task more complicated. This paper will investigate the effect of fine-tuning the recent pre-trained models such as BERT, XLNet, RoBERTa, and ELECTRA on multi-label patent classification. Comparison of these new models with the existing deep learning-based methods in the patent domain is worth investigating, and to the best of our knowledge, has not yet been done in this scope.

Pre-trained language models

In unsupervised representation learning, the models are first pre-trained on a large text corpus and then fine-tuned on some downstream task. One of the common unsupervised pre-training objectives in NLP is autoregressive (AR) language modeling. An AR-based language model is trained to predict the next word given a set of previous words or context in a uni-directional way (forward or backward). Particularly, given a text corpus with k words or tokens, $\mathbf{c} = (c_1, c_2, \dots, c_K)$, the traditional objective of language modeling is to estimate the joint probability $P(\mathbf{c})$, where this joint probability is usually auto-regressively factorized as shown in Eq. 1. Then a neural network is usually trained for modeling each conditional distribution (Dai et al., 2019; Yang et al., 2019).

$$P(\mathbf{c}) = \prod_{k=1}^K P(c_k | \mathbf{c}_{<k}) \quad (1)$$

The pre-training is achieved by maximizing the likelihood based on the forward autoregressive factorization as follows (Yang et al., 2019).

$$\max_{\theta} \log p_{\theta}(\mathbf{c}) = \sum_{k=1}^K \log p_{\theta}(c_k | \mathbf{c}_{<k}) = \sum_{k=1}^K \log \frac{\exp(h_{\theta}(\mathbf{c}_{1:k-1})^{\top} e(c_k))}{\sum_{c'} \exp(h_{\theta}(\mathbf{c}_{1:k-1})^{\top} e(c'))} \quad (2)$$

where $h_{\theta}(\mathbf{c}_{1:k-1})$ is a context vector representation provided by neural models, e.g., RNNs, and $e(c)$ indicates the embedding of c .

The Transformer architecture proposed by Vaswani et al. (2017), as a sequence transduction model, is the foundation of NLP's recent essential advancements. Unlike RNNs, which take words as input one by one in a sequential form, the Transformer is a self-attention based model that takes all the input words at once. Therefore, it achieves better performance by capturing better contextual representations of the text. The Transformer needs less training time due to implementing computations in parallel. BERT (Devlin et al., 2018), Bidirectional Encoder Representations from Transformers, uses the Transformer encoder architecture to learn language representations based on a bidirectional context. However, BERT uses another pre-training objective (an autoencoding-based pre-training) called *masked language modeling* (MLM) that makes this bidirectional pre-training possible. Given a text sequence $\mathbf{c}$, it corrupts the text by randomly replacing about 15% of the words or tokens in a sequence with a particular [MASK] token. Let the corrupted text and the masked tokens be $\mathbf{c}_{corrupt}$ and $\mathbf{c}_{masked}$, respectively. The pre-training objective is to reconstruct $\mathbf{c}_{masked}$ from $\mathbf{c}_{corrupt}$ as follows (Yang et al., 2019).

$$\begin{aligned} \max_{\theta} \log p_{\theta}(c_{masked} | c_{corrupt}) &\approx \sum_{k=1}^K m_k \log p_{\theta}(c_k | c_{corrupt}) \\ &= \sum_{k=1}^K m_k \log \frac{\exp(H_{\theta}(c_{corrupt})_k^{\top} e(c_k))}{\sum_{c'} \exp(H_{\theta}(c_{corrupt})_k^{\top} e(c'))} \end{aligned} \quad (3)$$

where $m_k = 1$ specifies that c_k is a masked token, and H_{θ} represents a Transformer that maps a sequence $\mathbf{c}$ of length K to series of hidden vectors, i.e., $H_{\theta}(\mathbf{c}) = (H_{\theta}(\mathbf{c})_1, \dots, H_{\theta}(\mathbf{c})_K)$.

The pre-trained model's parameters are utilized to initialize models to be fine-tuned on a variety of downstream tasks, which resulted in significant performance enhancement. However, BERT overlooks the dependency that exists between the masked positions. This is because it predicts all the masked positions in parallel at once. Moreover, BERT faces the problem of a pretrain-finetune discrepancy. The masked tokens or words that are the focus of the training stage do not appear when implementing fine-tuning on the downstream tasks.

XLNet (Yang et al., 2019), a generalized autoregressive pre-training model, was proposed to address the weak points of the BERT model. XLNet utilizes *permutation language modeling* instead of MLM, which also makes it possible to capture bidirectional context. For a text sequence $\mathbf{c}$ with length K , to do a valid autoregressive factorization, there are $K!$ distinct orders. Consider Q_k as the set of all the possible permutations of the index sequence of length K , $(1, 2, \dots, K)$. The permutation language modeling objective is shown in Eq. 4.

$$\max_{\theta} \mathbb{E}_{\mathbf{q} \sim Q_k} \left[\sum_{k=1}^K \log p_{\theta}(c_{q_k} | \mathbf{c}_{\mathbf{q}<k}) \right] \quad (4)$$

where $\mathbf{q} \in Q_k$ is a permutation, and q_k and $\mathbf{q}_{<k}$ represent the k -th and the first $k-1$ elements of the permutation. The factorization order $\mathbf{q}$ is sampled one at a time and the likelihood $p_{\theta}(\mathbf{c})$ is decomposed based on the factorization order. During training, the same parameter θ is shared among all the factorization orders, which makes the bidirectional

Fig. 1 An overview of RTD in ELECTRA (Clark et al., 2020)

context learning possible. Given the context, XLNet predicts the tokens in random order and not necessarily from left to right. XLNet is based upon Transformer-XL (Dai et al., 2019) architecture, not the vanilla Transformer, resulting in achieving better performance compared to the BERT model.

Moreover, Liu et al. (2019) proposed RoBERTa that altered the key hyperparameters in BERT and conducted pre-training longer on much larger batches, learning rates, and data. Compared to BERT, it improved the masked language modeling objective by dynamically altering the pattern of masking and achieved better performance on the downstream tasks.

On the other hand, ELECTRA (Clark et al., 2020) pre-training language model uses *replaced token detection (RTD)*, a new sample-efficient pre-training task, that learns a bidirectional model from all input positions. Figure 1 represents an overview of RTD in ELECTRA. As can be seen in the figure, it consists of two Transformer encoder models, a generator (G) and a discriminator (D), respectively. The G model is similar to BERT and is trained to perform masked language modeling similar to Eq. 3. The masked tokens are replaced with the samples from G and then fed into the D model. The discriminator then predicts whether each of the input tokens at position k was replaced by G or not with a sigmoid output layer as follows:

$$D(\mathbf{c}, k) = \text{sigmoid}(w^\top H_D(\mathbf{c})_k) \quad (5)$$

The model requires to learn an accurate representation of the data distribution for solving this task. Using all the input tokens in the RTD task makes ELECTRA more efficient. Therefore, resulting in a more powerful representation learning model. Given the same computation resource, model size, and data, ELECTRA can achieve the same performance by seeing fewer examples. Moreover, it gives comparable performance to the Roberta and XLNet model while utilizing less than 1/4 of their computation power.

Multi-label patent classification

Let $P = \{p_1, p_2, \dots, p_n\}$ be a collection of n patent documents. We consider each document as a sequence of k words or tokens, $p_i = (c_1, c_2, \dots, c_k)$. Patent classification is a multi-label classification problem. That is, a patent document $p_i \in P$ can be assigned to multiple labels $l_j \in L$ where $L = \{l_1, \dots, l_m\}$ is the set of m labels and $\text{label}(p) = \{l_x, \dots, l_z\}$ indicates the set of all the labels (subset of L) that patent document p belongs to. In this study we only consider the subclass level of the IPC taxonomy. Our goal is to train a function $f: P \rightarrow 2^m$ to map a patent document p to a set of correct labels $\text{label}(p)$.

Fig. 2 Preview of fine-tuning process using pre-trained language models

We propose using pre-trained language models for representing the patent text and fine-tuning them for the downstream task of the multi-label patent classification problem. In this paper, we utilize the pre-trained BERT, RoBERTa, XLNet, and ELECTRA language models. For adapting these models for patent classification, a linear layer (fully connected layer) with the size equal to m is added on the top of the language models. We need to fine-tune the pre-trained language model's parameters and train the newly added classifier end-to-end using patent data. Figure 2 depicts an overview of the fine-tuning process, which we will explain in more detail in the following.

The pre-processing step is needed to prepare the input text to match the pre-trained language models' expected input format. The general text pre-processing steps are as follows. First, we need to tokenize the input sequence. That is to split the raw text into words, subwords, punctuation symbols, or in general, smaller units (called tokens) that are in the pre-trained models' vocabulary or recognizable by the model. Second, the appropriate special tokens (e.g., the classification [CLS] and the separator [SEP] tokens) required when implementing text classification are added to the input sequences. The tokens are then converted to numerical values: their associated index in the pre-trained model's vocabulary or the embedding table. Moreover, the input sequence will be padded or truncated to a fixed length before feeding into the model. The fixed-length sequences will help with batching of the inputs and using GPUs. Concurrently, for preventing the model from performing attention on the padded tokens, an attention mask is also fed to the model (not shown in the Figure) that specifies the padded tokens. The attention mask values are 0 or 1, with 0 values corresponding to padded tokens.

BERT and ELECTRA use the WordPiece (Wu et al., 2010; Schuster & Nakajima, 2012) tokenization algorithm. In contrast, RoBERTa uses Byte-Pair Encoding (BPE) (Sennrich et al., 2015), and XLNet uses SentencePiece (Kudo & Richardson, 2018) combined with Unigram or BPE method for text tokenization and constructing their model vocabulary.

Furthermore, the IPC subclass labels associating with each patent text will be encoded. Given m unique labels, each document is associated with a list equal to the number of labels. Each position in the list represents a specific label and will have value one if that document belongs to that label and zero otherwise. Formally, the corresponding label vector for each p_i is converted to $l_i \in \{0, 1\}^m$, where $l_{ij} = 1$ if the i -th document belongs to the j -th label and $l_{ij} = 0$ otherwise.

Each pre-trained language model has a hidden size of H . That is, the input tokens of the models, when passed through the model, will be associated with an embedding vector of size H . The [CLS] token is inserted at the beginning of each input sequence, except in the XLNet model that is positioned at the end of the sequence. This essential [CLS] token that is showed in orange color in Fig. 2 is used as an aggregate representation for each patent document p_i . The input of the fully connected layer for each p_i is the embedding vector of the final hidden layer corresponding to [CLS] token, which we represent as $S_i \in \mathbb{R}^H$. The classification layer weights that we need to train are depicted as $W \in \mathbb{R}^{m \times H}$, where m is the number of labels.

However, instead of the original softmax cross-entropy loss function, which is appropriate for the multi-class classification problem, we consider the binary cross-entropy with logits loss function that suits multi-label classification (Liu et al., 2017). It converts the problem into a binary classification problem for each label independently, and therefore each document can have more than one label allocated to it. The logits indicate the raw outputs before it is fed into the sigmoid activation function ($v = SW^T \in \mathbb{R}^m$). Binary cross-entropy with logits loss can be formulated as follows.

$$-\frac{1}{n} \sum_{i=1}^n \sum_{j=1}^m [l_{ij} \log \sigma(v_{ij}) + (1 - l_{ij}) \log (1 - \sigma(v_{ij}))] \quad (6)$$

where $\sigma(y) = \frac{1}{1+e^{-y}}$ is the sigmoid function and v_{ij} is the logits or raw output scores of the fully connected layer. The sigmoid will reduce all the values between zero and one. Then a common approach is to consider a threshold value and convert the probabilities predicted for all the labels that are higher than the threshold to one and zero otherwise.

Experiment

In this section, first, we introduce the datasets used for conducting experiments. After that, the evaluation measures used are defined. The details of the baseline models and the experimental setting are provided next. Finally, we present experimental results and discussions.

Dataset

In this study, to be able to compare the experiments with previous works, two datasets are used: USPTO-2M and M-patent which we will describe in more detail as follows.

Table 1 Description of USPTO-2M

	Training data	Testing data
Number of documents	1,947,223	49,888
Average number of labels per patent	1.32	1.93
Number of IPC subclass level labels	544	544

Table 2 Description of M-patent

	Training data	Testing data
Number of documents	55,617	13,905
Average number of labels per patent	1.4	1.4
Number of IPC subclass level labels	96	96

USPTO-2M: The USPTO-2M is a large-scale patent classification benchmark made publicly available by Li et al. (2018)². The USPTO-2M is extracted from the bulk data available online on the United States Patent and Trademark Office (USPTO)’s website³. The preprocessed data includes the titles and abstract sections, patent number, and the IPC subclass labels of patent documents. The training data consist of 1,950,247 patents (from 2006–2014), and the test data contains 49,900 patents (2015), with 637 and 606 subclasses, respectively. However, after downloading the data and conducting some data analysis, we found that 1,739 of the training data had no labels. Using the patent number of the data with missing labels, we conducted web scraping from the USPTO’s website. Consequently, the missing labels of 1,719 of the data were extracted and replaced. Similar to Li et al., the documents with less than ten words are excluded. However, we also removed the labels that contained less than 100 documents. We only considered 544 labels at the IPC subclass level. The description of the train and test data are presented in Table 1. It should be noted that there is a slight difference between the dataset used in this paper and the data described in (Li et al., 2018). That is why, in our experiments, we implement their proposed method again on our dataset.

M-patent: Moreover, we used a smaller dataset, which is a subset of the CLEF-IP 2011 (Piroi et al., 2011) dataset. We call this dataset as M-patent dataset. The CLEF-IP dataset includes 1.35 million patents gathered from the European Patent Office (EPO)⁴ and World Intellectual Property Organization (WIPO)⁵. This dataset contains English, French, and German language patents. Similar to Hu et al. (2018a) and their M-CLEF dataset, the title, abstract, description, and claim part of English patents that belong to the F category of IPC taxonomy are extracted. The extracted documents have at least one IPC label, and the labels are chosen at the subclass level of the IPC hierarchy. After removing documents that were missing any of the desired parts and cleaning the data, the dataset includes 69, 522 data. Table 2 depicts the description of the train and test data.

² <http://mleg.cse.sc.edu/DeepPatent/>.

³ <https://www.uspto.gov/>.

⁴ <https://www.epo.org/>.

⁵ <https://www.wipo.int/portal/en/index.html>.

Evaluation measures

One of the problems of previous research on patent classification, as mentioned before, is that different evaluation measures are used when reporting the classification results. Unlike single-label classification, in multi-label classification, one or more labels can be assigned to an output concurrently. Therefore, evaluating the performance of the multi-label classification is harder and more complicated (Wu & Zhou, 2017), especially when the number of labels is high, like in the case of patent classification. Moreover, the distribution of the patent documents across the IPC categories is highly imbalanced (Gomez & Moens, 2014; Lupu et al., 2017). We believe not all the previous evaluation measures suit our problem. However, we try to report the evaluation measures used in previous patent research for the sake of comparability.

Multi-label evaluation measures can be divided into two main groups, namely, label-based metrics and example-based metrics (Tsoumakas et al., 2009). Example-based metrics are calculated independently for each example and then averaged on the total number of examples. In contrast, label-based metrics are calculated for each label independently and then averaged. Two averaging strategies can be utilized when using label-based metrics, namely, macro-averaging and micro-averaging (Wu & Zhou, 2017). Macro-averaging is similar to example-based averaging but for labels instead. On the other hand, in the micro-averaging strategy, the counters of misses and hits are aggregated first, and then the desired metric is calculated only once (Charte et al., 2016). Consequently, the weights assigned to each label in calculating the final measure are not the same, and the uneven distribution of data in each label is taken into consideration. Therefore, when we have imbalanced data, it is common to evaluate the performance with micro-*F1* measure (Yang et al., 2009).

Given n patents and m labels, let the number of true positives, false positives, true negatives, and false negatives, be TP , FP , TN , FN , respectively. The micro precision, recall, and *F1* measure (Gibaja & Ventura, 2014) are calculated as follows.

$$\text{Micro Precision} = \frac{\sum_{i=1}^m TP_i}{\sum_{i=1}^m TP_i + \sum_{i=1}^m FP_i} \quad (7)$$

$$\text{Micro Recall} = \frac{\sum_{i=1}^m TP_i}{\sum_{i=1}^m TP_i + \sum_{i=1}^m FN_i} \quad (8)$$

$$\text{Micro } F1 = \frac{2 \times \text{Micro Precision} \times \text{Micro Recall}}{\text{Micro Precision} + \text{Micro Recall}} \quad (9)$$

Moreover, we also report two ranking-based measures, Coverage Error (Wu & Zhou, 2017) and Label Ranking Average Precision (LRAP) (Tsoumakas et al., 2009). Given the ground truth label matrix as $L \in \{0, 1\}^{n \times m}$, let the probabilities estimated by the model before converting to binary bipartition using a threshold value be $\hat{L} \in \mathbb{R}^{n \times m}$. Coverage error indicates, on average, how many labels in the ranked list of the estimated probabilities are required to account for all the true positive labels and is computed as follows:

$$\text{Coverage Error} = \frac{1}{n} \sum_{i=1}^n \max_{j: \hat{l}_{ij}=1} \text{rank}_{ij} \quad (10)$$

where $\text{rank}_{ij} = \left| \{g : \hat{l}_{ig} \geq \hat{l}_{ij}\} \right|$.

Fig. 3 Number of words statistics of the datasets

LRAP is related to the average precision score. However, it uses the concept of label ranking instead of precision and recall. It evaluates the capability of the classifier to assign better ranks to the correct labels associated with each sample and is calculated as follows:

$$LRAP = \frac{1}{n} \sum_{i=1}^n \frac{1}{\|l_i\|_0} \sum_{j:l_{ij}=1} \frac{|L_{ij}|}{rank_{ij}} \quad (11)$$

where $L_{ij} = \{g : l_{ig} = 1, \hat{l}_{ig} \geq \hat{l}_{ij}\}$, $|L|$ is the cardinality of a set, and $\|\cdot\|_0$ is the l_0 -norm.

Some previous research (Li et al., 2018; Hu et al., 2018a) utilized the evaluation measures from the CLEP-IP competition (Piroi et al., 2011). That is, first predict k (e.g., 1, 5) labels for each document, and then calculate the precision, recall, and $F1$ measure at top- k for each prediction as shown below. For the sake of comparison, we will report the precision, recall, and $F1$ at the top 1 label.

$$\text{Precision} = \frac{\text{correct predictions}}{\text{all predictions}} \quad (12)$$

$$\text{Recall} = \frac{\text{correct predictions}}{\text{all relevant documents}} \quad (13)$$

$$F1 \text{ measure} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \quad (14)$$

For all the metrics, a higher obtained value is better, except for coverage error, in which a smaller value is preferable.

Baselines and experiment setup

Li et al. (2018) evaluated the effect of using different patent sections and the number of words on the classification performance on the USPTO-2M dataset. They concluded that using the first 100 words of the title and abstract will result in the best classification performance. Therefore, to compare our experiments with the DeepPatent model proposed by Li

et al. (2018), we will use the first 100 words of the title and abstract for our experiments on the USPTO-2M dataset. Figure 3a shows the number of words in the title and abstract sections of the USPTO-2M dataset.

For the M-patent dataset, we follow the same guideline and select the title and abstract sections. As shown in Fig. 3 (b), about 95% of the combination of title and abstract in the M-patent dataset contains less than 128 words (red line drawn on 128). Consequently, we set the maximum number of words to 128 in the corresponding experiments. Moreover, the standard 80/20 split was applied for the training and validation set. Similar to Hu et al. (2018a), we further employ other deep learning baseline models used for classification, namely, LSTM, BiLSTM, CNN and CNN-BiLSTM. We keep the input text the same for all the models to make a fair comparison.

We describe the baseline models first and then explain the fine-tuning pre-trained language models' implementation details for the patent classification.

DeepPatent: We used a CNN architecture similar to DeepPatent (Li et al., 2018) and trained 200-dimensional word embeddings based on the skip-gram model. The CNN architecture is as follows: a convolutional layer with three kernel sizes (3, 4, and 5), a max-pooling layer applied to the output of each convolutional layer, and a fully connected layer with units equal to the number of labels and sigmoid activation function. Furthermore, we also examined 200-dimensional word embeddings trained with the CBOW and fastText model.

LSTM: A single layer LSTM with 128 units. Followed by a fully connected layer in which the number of units is equal to the number of labels and sigmoid functions.

BiLSTM: The same architecture with the LSTM model, but the LSTM units, is replaced with the BiLSTM units.

CNN-BiLSTM- A convolutional layer with kernel size 3, rectified linear units (ReLU) activation function, and 128 kernels. The global max-pooling operation was applied to obtain the feature vectors of the input. The feature maps are then fed into a BiLSTM layer with 128 units. Dropout is applied to the output of BiLSTM layer and fed into a fully connected layer with the number of units equal to the number of labels and sigmoid activation functions.

For all the models, Adam (Kingma & Ba, 2014) optimizer and a binary cross-entropy loss function were used. Moreover, we use dropout (Srivastava et al., 2014) after the embedding layer and also before the fully connected layer with a drop rate of 0.2 and 0.25, respectively. Similar to Li et al. (2018), we train DeepPatent for 50 epochs on the USPTO-2M dataset. For the experiments on the M-patent dataset, the number of epochs is set to 40 same as Hu et al. (2018a).

For converting the input text to a sequence of tokens, padding, or truncating to the desired length and implementing the baseline models, Keras library Chollet et al. (2015) was utilized. Each word of the input sequence is represented with a 200 dimension word vector. We tested three different word vectors generated based on CBOW, Skipgram, and fastText for all the models.

Word embedding pre-training: Two common word embedding models, i.e., word2vec, and fastText, are used to generate 200-dimensional word embeddings using the patent texts in our datasets. The Python Gensim library (Řehůřek & Sojka, 2010) is utilized to generate word2vec embeddings based on both the CBOW and Skip-gram model, and the fastText embeddings based on the skip-gram model with a context window size of 5. We implemented the standard preprocessing steps such as removing punctuation, non-alphabetic characters, stop words, converting the text to lowercase, lemmatization, and reducing all the multi-spaces to single space.

Table 3 Pre-trained models name and details

Model	Model size	Model details
BERT	bert-base-uncased	12-layer, 768-hidden, 12-heads, 110M parameters
XLNet	xlnet-base-cased	12-layer, 768-hidden, 12-heads, 110M parameters
RoBERTa	roberta-base	12-layer, 768-hidden, 12-heads, 125M parameters
ELECTRA	ELECTRA-Base	12-layer, 768-hidden, 12-heads, 110M parameters

Table 4 Hyperparameter setting for fine-tuning various pre-trained models

Hyperparameter	Value
Batch size	64
Number of epochs	4
Maximum sequence length	128
Learning rate	$3e^{-5}$

Fine-tuning pre-trained language models: BERT, XLNet, RoBERTa, and ELECTRA models are fine-tuned on the downstream task of multi-label patent classification problem. The publicly available Simple Transformers (Rajapakse, 2019) library built on top of the famous Hugging Face transformers library (Wolf et al., 2019) was used for conducting the experiments. The pre-trained model weights are all provided in the Hugging Face transformers library. The base versions of the models were used in our experiments. Table 3 shows the model types and their details.

An advantage of using such libraries is that we can use the pre-trained models in a unified way and conduct pre-training with limited computation resources. Only a few hyperparameters such as batch size, number of epochs, maximum sequence length, and the learning rate were changed for conducting the experiments. These hyperparameters were chosen based on the original paper recommendations and kept the same for all the models. Table 4 shows the hyperparameters in the experimental setting. However, for the experiment on the M-patent dataset we set the number of epochs to 15 with early stopping (Caruana et al., 2001) of patience level 3 to prevent over-fitting.

All the experiments are carried out on a 180GB RAM operating machine with Intel(R) Xeon(R) Gold 5120 CPU @ 2.20GHz, 16 cores and two Tesla V100 GPUs with 16 GB of RAM.

Experimental results and discussion

This section presents the experimental results on the two datasets, M-patent and USPTO-2M, in Tables 5 and 6, respectively. We fine-tuned various pre-trained language models for the task of multi-label patent classification and compared them with several neural network architectures proposed in the literature.

Table 5 demonstrates the result of experiments on the M-patent dataset. As can be seen from the table, all the pre-trained language models showed better classification performance by obtaining higher micro-F1 and F1 at the top 1 label prediction. This demonstrates the effectiveness of fine-tuning these models for multi-label patent classification and their ability to capture the patent documents features better than the

Fig. 4 Performance of the pre-trained language models on M-patent evaluation dataset

Table 5 Experiments on M-patent dataset

Methods	Top 1 (%)			Micro			Coverage Error	LRAP
	Precision	Recall	F1	Precision	Recall	F1		
LSTM (CBOW)	74.32	60.81	64.86	0.815	0.521	0.636	4.158	0.787
LSTM (Skipgram)	74.10	60.67	64.69	0.810	0.521	0.634	4.157	0.788
LSTM (fastText)	74.30	60.77	64.83	0.819	0.517	0.634	4.144	0.787
BiLSTM (CBOW)	77.11	63.22	67.39	0.801	0.585	0.676	3.549	0.816
BiLSTM (Skipgram)	77.99	64.03	68.22	0.817	0.577	0.677	3.386	0.823
BiLSTM (fastText)	78.25	64.21	68.42	0.817	0.585	0.681	3.431	0.824
CNN-BiLSTM (CBOW)	77.04	63.18	67.34	0.793	0.589	0.676	3.686	0.813
CNN-BiLSTM (Skip-gram)	78.66	64.55	68.78	0.810	0.598	0.688	3.460	0.828
CNN-BiLSTM (fastText)	79.30	65.13	69.38	0.821	0.593	0.689	3.337	0.831
DeepPatent (CBOW)	76.69	62.75	66.93	0.755	0.607	0.673	3.608	0.811
DeepPatent (Skip-gram)	79.68	65.47	69.73	0.791	0.621	0.696	3.164	0.837
DeepPatent (fastText)	79.77	65.52	69.79	0.787	0.627	0.698	3.131	0.837
BERT	81.49	66.96	71.33	0.797	0.673	0.730	4.102	0.841
XLNet	82.29	67.70	72.08	0.793	0.686	0.736	3.635	0.850
RoBERTa	80.87	66.43	70.77	0.813	0.643	0.718	3.834	0.836
ELECTRA	80.40	66.07	70.38	0.800	0.649	0.717	4.163	0.830

Table 6 Experiments on the USPTO-2M dataset

Methods	Top 1 (%)			Micro			Coverage Error	LRAP
	Precision	Recall	F1	Precision	Recall	F1		
DeepPatent (CBOW)	75.82	51.11	57.95	0.712	0.389	0.503	11.432	0.751
DeepPatent (Skip-gram)	77.10	52.00	58.95	0.810	0.386	0.523	10.732	0.763
DeepPatent (fastText)	78.03	52.63	59.67	0.809	0.402	0.537	10.311	0.771
BERT	82.43	55.69	63.11	0.857	0.429	0.572	9.845	0.803
XLNet	82.72	55.89	63.33	0.860	0.429	0.572	8.986	0.808
RoBERTa	82.42	55.68	63.10	0.862	0.419	0.564	9.426	0.805
ELECTRA	82.08	55.46	62.84	0.863	0.417	0.562	9.948	0.799

baseline models. XLNet outperformed all the other models and achieved the highest value of micro-F1 of 0.736 and 0.850 of LRAP. Moreover, the precision, recall, and F1 at the top 1 prediction are 82.29%, 67.70%, and 72.08%, respectively. Among the baseline models, except when using the CBOW word embeddings, DeepPatent obtained the best performance, followed by the CNN-BiLSTM model. DeepPatent with the fastText embeddings achieved 79.77%, 65.52%, and 69.79% of precision, recall, and F1 at the top 1 label prediction, respectively. On the other hand, LSTM models resulted in the lowest micro-F1 and other evaluation measures. The BiLSTM models, which can capture bidirectional dependencies in the text, depict better performance than the LSTM models. In general, the models that include LSTM or BiLSTM achieve higher micro-precision

Fig. 5 Comparison of various threshold values for XLNet model on USPTO-2M dataset

Fig. 6 Comparison of various threshold values for XLNet on M-patent

but lower recall. In the case of CBOW embeddings, adding CNN to the BiLSTM model does not introduce performance improvement. Figure 4 presents the performance of the pre-trained models on the M-patent validation set. The difference in the training steps is due to the stopping operation to prevent the models from over-fitting. None of the models exceeded training more than 11 epochs. The global training steps versus the relative time is shown in Fig. 5c, which indicates that XLNet took the longest to train among the pre-trained language models.

Furthermore, we also conducted experiments on the USPTO-2M dataset contributed by Li et al. (2018) and used their DeepPatent as the baseline model. Table 6 demonstrates the result of experiments on the USPTO-2M dataset. Implementing DeepPatent again on the dataset and using our trained Skip-gram word vectors, we obtained a precision, recall, and F1 of 77.10%, 52.00%, and 58.95% at the top 1 label prediction, respectively. With the usage of our trained fastText embeddings instead of word2vec embeddings, we outperformed the original DeepPatent model, resulting in better performance than the original

model. However, once again, all the pre-trained language models defeated DeepPatent in terms of all metrics. XLNet achieves a new state-of-the-art performance of 82.72%, 55.89%, and 63.33% on precision, recall, and F1 at the top 1 label prediction, respectively. Moreover, the micro-F1 of 0.523 obtained from the original DeepPatent (Skip-gram) increased to 0.572 using the XLNet model. XLNet also obtained the best LARP (0.808) and coverage error (8.986) on the USPTO-2M dataset.

The precision, recall, and F1 measures are reported by converting the models' output probabilities to binary bipartition vectors. The common approach is to consider a threshold value that transforms all the probabilities higher than the threshold to 1 and 0 otherwise. The results reported in Tables 5 and 6 are all based on considering the conventional 0.5 as the threshold value. However, in tasks when recall or precision is preferred to another, one might consider a different threshold value. The recall is considered more important in many patent-related tasks. Figures 5 and 6 illustrate how various threshold values change the micro-precision, recall, and F1 measure for the XLNet model (as an example) on the USPTO-2M and M-patent experiments.

Moreover, we investigated the effect of using the same length of inputs (the first 128 words) from the beginning of the description and the first claim of the M-patent dataset on the classification performance. Even though the micro-precision increases when using the beginning of the description, the overall micro F1 is not improved. The title and abstract sections resulted in better classification performance, as shown in Fig. 7. This is because the title and abstract sections are more informative than the description or claim for patent classification. However, a longer text might need to be considered when using the description or claim sections of the patent. Nevertheless, which part of the patent results in a better classification performance is not the focus of this study.

Overall, among the baseline deep learning models, the ones using the trained fastText embeddings on the patent corpus show better performance than word2vec embeddings trained based on the CBOW and Skip-gram algorithm. This is because the patent text is not the same as other scientific or academic text. The jargon, complex, and new technical terms abundant in patent text cause out-of-vocabulary problems when using conventional word embeddings. However, fastText can overcome the out-of-vocabulary problem by breaking the unseen word to character n-grams and sums the n-gram embeddings to provide a vector representation for the unseen word. The reason Skip-gram shows better performance than CBOW is that the Skip-gram model is better in learning from smaller data and representing rare words than the CBOW method. However, all of the embedding methods have the limitation of not being able to account for polysemy. Regardless of the context, each word is assigned a specific embedding vector. The ability to account for polysemy is important for understanding the patent context and extracting features from it. In patent documents, the same word may have different meanings in different technology areas. The pre-trained language models use embedding methods that can deal with out-of-vocabulary words and overcome the polysemy problem by pre-training contextual representations of the input text.

For encoding higher-level features from the word embeddings, even though LSTM is known to encode long-term dependencies in the text and to capture the sequential nature in a language, it is not powerful enough to capture the patent documents' context well for the multi-label classification task. The recurrent neural network-based models might particularly encode information that might not be entirely relevant for the classification task. This problem mainly occurs when the input text is long and very information-rich such as patent, where assigning relevant labels requires a more selective encoding. The Transformers' attention mechanism of the pre-trained language models is more powerful in capturing

Fig. 7 Performance comparison of the models using different sections of the patent on the M-patent dataset

the dependencies in a sequence than LSTM or even Bi-LSTM architectures. The LSTM or Bi-LSTM models are not powerful enough to semantically understand the patent text to assign all the associate labels to the patent document. DeepPatent is a CNN model. CNN

model is hierarchical, and different kernel sizes can extract various important n-gram features or semantic clues from the input. Therefore, it can capture the phrasal expressions in the patent data for the classification task but fails to model the long-distance dependencies or contextual information. The multi-word units or phrasal expressions contribute to identifying the related categories for each document. However, the general terms in the multi-word combination and the complexity of the patent documents make only relying on these n-gram features not sufficient for the multi-label patent classification task. Nevertheless, the deep Transformer layers of the pre-trained BERT, XLNet, RoBERTa, and ELECTRA, when fine-tuned on the patent text, can encode much richer patent document features considering the bi-directional context and lead to better classification performance.

The experimental results on both M-patent and USPTO-2M indicate that XLNet obtained the best performance on the multi-label patent classification task. One feasible explanation of this outcome is the permutation language modeling objective used in XLNet for pre-training. Except for XLNet, which is a generalized autoregressive pre-training model, other models are considered as autoencoding, i.e., they rely on somehow corrupting the input text and then try to restore it. However, the permutation language modeling in XLNet combines the benefits of both autoencoding and autoregressive methods while avoiding their shortcomings. For example, BERT uses the MLM pre-training objective to make bidirectional pre-training possible but fails to consider the dependency between the masked positions. The data corruption of masking tokens in BERT also leads to a potential pretrain-finetune discrepancy problem. However, the permutation language modeling objective of XLNet avoids these limitations and is more powerful by capturing more dependencies and phrasal structures available in the patent text compared to other pre-trained models. Furthermore, unlike other models that are based on the vanilla Transformer architecture, XLNet adopts the relative positional encoding scheme and the segment recurrence mechanism of the Transformer-XL. Therefore, XLNet can learn dependency better for tasks with longer text sequence. Even though RoBERTa and ELECTRA also improved the MLM objective of BERT, they still did not show better performance when fine-tuned on patent data. RoBERTa is technically similar to BERT, but it improves the MLM by dynamically altering the token masking pattern and conducted pre-training longer on much larger data. However, RoBERTa shows similar performance to BERT on the patent data. The classification results were even slightly better for BERT. This means that adding more data for pre-training does not necessarily mean better performance when fine-tuned on the patent text. Moreover, the RTD pre-training objective in ELECTRA also does not show any improvement over other models on the downstream task of patent classification. Consequently, XLNet, by having the additional autoregressive features, is more suitable for understanding and encoding the phrasal structures and the complex language used in patent documents and provides the best patent classification result among the pre-trained language models.

However, the pre-trained language models take longer to fine-tune, and even though XLNet obtained the best performance with the fixed hyperparameters, it took twice as much time to fine-tune compared to RoBERTa and ELECTRA model. On the USPTO-2M dataset, XLNet took around 22 hours and a half to fine-tune, while RoBERTa and ELECTRA took only around 10 hours and twenty minutes and 10 hours, respectively. CNN models are faster to train than other models and can achieve somewhat acceptable performances. Nonetheless, not much pre-processing of data is needed when using the pre-trained models. Even though we kept the hyperparameters the same for all of the pre-trained models for the sake of comparison, it still led to great results. Separately conducting systematic hyperparameter tuning for each model may lead to better performance, which we leave for future

research. Therefore, using the pre-trained language models is promising for enhancing patent classification performance and other patent analysis related tasks that require powerful language understanding models. Moreover, due to the unique characteristics of patent text and its importance, recently, Srebrovic and Yonamine (2020) from Google trained the BERT model, with a slight modification, exclusively on more than 100 million patent documents. They considered 512 input sequence size and limited the maximum masked words to 45 for a sequence. They used a custom tokenizer specifically optimized on the patent text that extends the standard BERT vocabulary to include frequently occurred words in patents. Therefore, preventing the long words that are more common in the patent text from breaking down into smaller word pieces, which should result in performance enhancement of downstream patent-related tasks. The main focus of their work was mainly to show how to utilize this model for contextual synonym generation in patents and its effectiveness. They also highlighted additional applications for the general classification and patent auto-complete tasks. The model and checkpoints, along with the configuration and vocab files, are publicly available at the Google patents-public-data GitHub repository⁶. However, the large BERT model consists of 24 layer Transformers, 1024 hidden dimensions, 16 attention heads, with approximately 340 million parameters, which is much larger than the base version of the BERT model. This significant number of parameters requires substantial computational resources and memory to even perform a single inference pass and to adopt this model for the classification task. Therefore, we could not include the Bert for patents in our experiments due to the limitations in computational resources. Additional scaling and speeding techniques may be required to deploy the BERT for patents model with less computational power, which we leave for future study.

Conclusions

Patent classification is one of the fundamental steps for managing patents and maintaining inter-operability among patent offices worldwide. Patents are classified based on some complex hierarchical standard taxonomies that are prone to change as time goes on. Moreover, patents can be assigned to one or more labels simultaneously. The unique structure and the characteristics of the patent text make patent retrieval, information mining, and effective automatic classification more challenging.

In this paper, the multi-label patent classification task's overall performance was improved using the pre-trained language models. We examined the pre-trained BERT, XLNet, RoBERTa, and ELECTRA models and fine-tuned them for multi-label patent classification. We compared these models with other deep learning-based models proposed for patent classification, namely, CNN, LSTM, BiLSTM, and CNN-BiLSTM. Three kinds of word embeddings based on CBOW, Skipgram, and fastText models were trained on patent data and investigated on the baseline models. The USPTO-2M and M-patent datasets on the IPC subclass were used for conducting experiments. We found that utilizing fastText embeddings shows better performance than word2vec embeddings on the baseline models. Overall, pre-trained language models provided better classification performance. Experiments conclude that XLNet outperformed all the models and achieved a new

⁶ <https://github.com/google/patents-public-data/blob/master/models/BERT%20for%20Patents.md>.

state-of-the-art result. To the best of our knowledge, this paper is the first to conduct such a thorough investigation utilizing the latest pre-trained language models.

The emergence of new language representation models, such as OpenAI GPT-3 and many more to come, are making the fine-tuning approach more efficient and easier to implement. Pre-trained language models are very powerful in language understanding and representing text and should be utilized for developing new systems in the patent domain. BERT for patents is also a promising pre-trained model, exclusively trained on a large patent corpus that needs to be included in more patent-related tasks. Our work's future direction is to take advantage of the pre-trained language models for developing a new system that can capture the hierarchical structure of the classification taxonomies. Furthermore, the effective classification of documents to lower IPC levels and the ability to account for the imbalanced nature of the patent data is still an open area of research. Future research on patents needs to utilize appropriate evaluation measures that suit the multi-label, hierarchical, and imbalanced nature of the data.

Acknowledgements This work was supported by the Ministry of Education of the Republic of Korea with the National Research Foundation of Korea (NRF-2019S1A5C2A03081234) and Inha University.

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

References

- Abdelgawad, L., Kluegl, P., Genc, E., Falkner, S., & Hutter, F. (2019). Optimizing neural networks for patent classification. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases* (pp. 688–703). Springer.
- Al Shamsi, F., & Aung, Z. (2016). Automatic patent classification by a three-phase model with document frequency matrix and boosted tree. In *2016 5th International Conference on Electronic Devices, Systems and Applications (ICEDSA)* (pp. 1–4). IEEE.
- Bojanowski, P., Grave, E., Joulin, A., & Mikolov, T. (2017). Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5, 135–146.
- Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al. (2020). Language models are few-shot learners. arXiv preprint [arXiv:2005.14165](https://arxiv.org/abs/2005.14165)
- Caruana, R., Lawrence, S., & Giles, C. L. (2001). Overfitting in neural nets: Backpropagation, conjugate gradient, and early stopping. In *Advances in Neural Information Processing Systems* (pp. 402–408).
- Charte, F., del Jesus, M. J., & Rivera, A. J. (2016). *Multilabel classification: Problem analysis, metrics and techniques*. Berlin: Springer.
- Chen, J., Chen, J., Zhao, S., Zhang, Y., & Tang, J. (2020). Exploiting word embedding for heterogeneous topic model towards patent recommendation. *Scientometrics*, 125(3), 2091–2108.
- Chen, L., Xu, S., Zhu, L., Zhang, J., Lei, X., & Yang, G. (2020). A deep learning based method for extracting semantic information from patent documents. *Scientometrics*, 125(1), 289–312.
- Chollet, F., et al. (2015). Keras. <https://github.com/fchollet/keras>.
- Clark, K., Luong, M. T., Le, Q. V., & Manning, C. D. (2020). Electra: Pre-training text encoders as discriminators rather than generators. arXiv preprint [arXiv:2003.10555](https://arxiv.org/abs/2003.10555).
- Dai, Z., Yang, Z., Yang, Y., Carbonell, J., Le, Q. V., & Salakhutdinov, R. (2019). Transformer-xl: Attentive language models beyond a fixed-length context. arXiv preprint [arXiv:1901.02860](https://arxiv.org/abs/1901.02860).

- Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint [arXiv:1810.04805](https://arxiv.org/abs/1810.04805).
- D'hondt, E., & Verberne, S. (2010). Clef-ip 2010: Prior art retrieval using the different sections in patent documents.
- D'hondt, E., Verberne, S., Koster, C., & Boves, L. (2013). Text representations for patent classification. *Computational Linguistics* 39(3), 755–775.
- D'hondt, E., Verberne, S., Oostdijk, N., & Boves, L. (2017). Patent classification on subgroup level using balanced winnow. In *CCPIR* (pp. 299–324). Springer.
- Fall, C. J., Töröcsvári, A., Benzineb, K., & Karetka, G. (2003). Automated categorization in the international patent classification. In *Acm Sigir Forum, ACM New York, NY, USA* (Vol. 37, pp. 10–25).
- Gibaja, E., & Ventura, S. (2014). Multi-label learning: a review of the state of the art and ongoing research. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 4(6), 411–444.
- Gomez, J. C. (2019). Analysis of the effect of data properties in automated patent classification. *Scientometrics*, 121(3), 1239–1268.
- Gomez, J. C., & Moens, M. F. (2014). A survey of automated hierarchical classification of patents. In *PSMW* (pp. 215–249). Springer.
- Grawe, M. F., Martins, C. A., & Bonfante, A. G. (2017). Automated patent classification using word embedding. In *2017 16th IEEE International Conference on Machine Learning and Applications (ICMLA)* (pp. 408–411). IEEE.
- Hu, J., Li, S., Hu, J., & Yang, G. (2018). A hierarchical feature extraction model for multi-label mechanical patent classification. *Sustainability*, 10(1), 219.
- Hu, J., Li, S., Yao, Y., Yu, L., Yang, G., & Hu, J. (2018). Patent keyword extraction algorithm based on distributed representation for patent classification. *Entropy*, 20(2), 104.
- Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. arXiv preprint [arXiv:1412.6980](https://arxiv.org/abs/1412.6980).
- Kudo, T., & Richardson, J. (2018). Sentencepiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. arXiv preprint [arXiv:1808.06226](https://arxiv.org/abs/1808.06226).
- Lee, J. S., & Hsiang, J. (2019). Patentbert: Patent classification with fine-tuning a pre-trained bert model. arXiv preprint [arXiv:1906.02124](https://arxiv.org/abs/1906.02124).
- Li, S., Hu, J., Cui, Y., & Hu, J. (2018). Deeppatent: patent classification with convolutional neural networks and word embedding. *Scientometrics*, 117(2), 721–744.
- Liu, J., Chang, W. C., Wu, Y., & Yang, Y. (2017). Deep learning for extreme multi-label text classification. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval* (pp. 115–124).
- Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., & Stoyanov, V. (2019). Roberta: A robustly optimized bert pretraining approach. arXiv preprint [arXiv:1907.11692](https://arxiv.org/abs/1907.11692).
- Lupu, M., & Hanbury, A. (2013). Patent retrieval. *Foundations and Trends in Information Retrieval*, 7(1), 1–97.
- Lupu, M., Mayer, K., Kando, N., & Trippe, A. J. (2017). *Current challenges in patent information retrieval* (Vol. 37). Berlin: Springer.
- Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). Efficient estimation of word representations in vector space. arXiv preprint [arXiv:1301.3781](https://arxiv.org/abs/1301.3781).
- Minaee, S., Kalchbrenner, N., Cambria, E., Nikzad, N., Chenaghlu, M., & Gao, J. (2020). Deep learning based text classification: A comprehensive review. arXiv preprint [arXiv:2004.03705](https://arxiv.org/abs/2004.03705).
- Pennington, J., Socher, R., & Manning, C. D. (2014). Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)* (pp. 1532–1543).
- Piroi, F., Lupu, M., Hanbury, A., & Zenz, V. (2011). Clef-ip 2011: Retrieval in the intellectual property domain. In *CLEF (notebook papers/labs/workshop)*.
- Rajapakse, T. (2019). Simple transformers. <https://github.com/ThilinaRajapakse/simpletransformers>.
- Řehůřek, R., & Sojka, P. (2010). Software framework for topic modelling with large corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks, ELRA, Valletta, Malta* (pp. 45–50). <http://is.muni.cz/publication/884893/en>.
- Risch, J., & Krestel, R. (2019). Domain-specific word embeddings for patent classification. *Data Technologies and Applications*.
- Roudsari, A. H., Afshar, J., Lee, S., & Lee, W. (2021). Comparison and analysis of embedding methods for patent documents. In *2021 IEEE International Conference on Big Data and Smart Computing (BigComp)* (pp. 152–155). <https://doi.org/10.1109/BigComp51126.2021.00037>.

- Schuster, M., & Nakajima, K. (2012). Japanese and korean voice search. In *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (pp. 5149–5152). IEEE.
- Sennrich, R., Haddow, B., & Birch, A. (2015). Neural machine translation of rare words with subword units. arXiv preprint [arXiv:150807909](https://arxiv.org/abs/1508.07909).
- Shalaby, M., Stutzki, J., Schubert, M., & Günnemann, S. (2018). An lstm approach to patent classification based on fixed hierarchy vectors. In *SIAM* (pp. 495–503). SIAM.
- Shalaby, W., & Zadrozny, W. (2019). Patent retrieval: a literature review. *Knowledge and Information Systems*, 1–30.
- Song, J. J., Lee, W., & Afshar, J. (2019). An effective high recall retrieval method. *Data & Knowledge Engineering*, 123, 101603.
- Souza, C. M., Meireles, M. R., & Almeida, P. E. (2020). A comparative study of abstractive and extractive summarization techniques to label subgroups on patent dataset. *Scientometrics*, 1–22.
- Srebrovic, R., & Yonamine, J. (2020). Leveraging the bert algorithm for patents with tensorflow and bigquery [white paper]. https://services.google.com/fh/files/blogs/bert_for_patents_white_paper.pdf.
- Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1), 1929–1958.
- Tsoumakas, G., Katakis, I., & Vlahavas, I. (2009). Mining multi-label data. In *Data mining and knowledge discovery handbook* (pp. 667–685). Springer.
- Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., & Polosukhin, I. (2017). Attention is all you need. arXiv preprint [arXiv:170603762](https://arxiv.org/abs/1706.03762).
- Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., Cistac, P., Rault, T., Louf, R., Funtowicz, M., et al. (2019). Huggingface's transformers: State-of-the-art natural language processing. [arXiv:191003771](https://arxiv.org/abs/1910.03771).
- Wu, C. H., Ken, Y., & Huang, T. (2010). Patent classification system using a new hybrid genetic algorithm support vector machine. *Applied Soft Computing*, 10(4), 1164–1177.
- Wu, J. L., Chang, P. C., Tsao, C. C., & Fan, C. Y. (2016). A patent quality analysis and classification system using self-organizing maps with support vector machine. *Applied Soft Computing*, 41, 305–316.
- Wu, X. Z., & Zhou, Z. H. (2017). A unified view of multi-label performance measures. In *International Conference on Machine Learning, PMLR* (pp. 3780–3788).
- Yang, B., Sun, J. T., Wang, T., & Chen, Z. (2009). Effective multi-label active learning for text classification. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining* (pp. 917–926).
- Yang, Z., Dai, Z., Yang, Y., Carbonell, J., Salakhutdinov, R., & Le, Q. V. (2019). Xlnet: Generalized autoregressive pretraining for language understanding. arXiv preprint [arXiv:190608237](https://arxiv.org/abs/1906.08237).
- Young, T., Hazarika, D., Poria, S., & Cambria, E. (2018). Recent trends in deep learning based natural language processing. *CIM*, 13(3), 55–75.
- Yun, J., & Geum, Y. (2020). Automated classification of patents: A topic modeling approach. *Computers & Industrial Engineering*, 147, 106636.
- Zhang, L., Li, L., & Li, T. (2015). Patent mining: a survey. *SIGKDD Explorations*, 16(2), 1–19.