THE OHIO STATE UNIVERSITY

Multi-label classification of Patent Text for Efficient Information Retrieval

Project Category: Text
Physics 5680, Autumn 2024

Author: Ashley Tarrant

December 10, 2024

Abstract

Patents play a critical role in innovation and economic development by protecting inventor rights. For patents to be granted, they must be categorized into Cooperative Patent Classification (CPC) classes according to their technical nature for future retrieval and infringement claims. However, this process is difficult and time-consuming as the CPC assignment is determined by a domain-expert and patents can belong to multiple classes. To eliminate ambiguity in the assignment of CPC classes and increase efficiency in granting patents, we created a neural network to predict one or more CPC classes based on the title and abstract of US patents. We use PatentsBERTa, a text embedding model trained on US patents, to generate embeddings for patent text and predict the one-hot-encoded CPC class(es) the text corresponds to. We train three different networks, a fully connected network (FCN), an FCN with class weights, and a 1D convolutional neural network, finding that the CNN outperforms both FCNs. We achieve a binary accuracy of 99% while training, but our models still struggle overall to make true positive predictions given a macro-average F1-score of 0.34. Thus, this work is a stepping stone towards accurate and efficient patent classification for individuals, corporations, and patent reviews, supporting innovation in the US economy.

1 Introduction

Patents play a key role in economic development by protecting the rights of inventors to be the sole reproducer and seller of new technologies and products [1]. By offering incentives to develop new technologies, patents encourage innovative products and services which widely benefit society. A crucial step in the patent-granting process is classification, in which the patents are given a hierarchical section, class, and group to facilitate the retrieval of technical information [2]. Classification is critical for analysis of the patent content both before the patent is granted for future informational retrieval: to compare with new technologies and evaluate infringement of new patents. Previously, this task has been completed by domain experts, but as the volume and complexity of patent applications continues to increase each year, automation is necessary to keep up [3]. Employing machine learning techniques to directly classify patents based on their content will increase efficiency of the patent-granting process by significantly reducing the labor required for classification, search and retrieval, and a number of related support tasks.

Machine learning has demonstrated great potential in natural language processing (NLP) through analysis of text, the predominant component in patent applications. Therefore, training language models is a natural and

efficient method to classify patents directly from their content. Text can be converted to machine-readable data through embeddings, a multi-dimensional vector representation of the text and semantic information. We then train three different multi-label neural networks which predicts the Cooperative Patent Classification (CPC) class(es) corresponding to the patent text embeddings. Thus our method will assist patent examiners in the pre- and post-processing phases of patent granting, overall increasing the efficiency of this process.

2 Related Work

Several studies have employed machine learning techniques to increase the efficiency and accuracy of patent classification. One network designed to accomplish this task is PatentNet, a fine-tuned version of the XLNet NLP model trained specifically to classify patent text [3]. This study thoroughly examines several pre-trained language models with different word embeddings and were able to produce state-of-the-art classification results with the reduced cost of employing pre-trained models. This shows promising potential as more large language models (LLM) continue to emerge such as OpenAI's ChatGPT and BERT [4]. Although producing promising results, fine-tuning their model requires substantial computational resources, taking over 22 hours, while other architectures such as CNNs take significantly less time to train. Another proposed network, DeepPatent [5], capitalizes on the ability of convolutional neural networks to learn important features from input data to generate a multi-class, multi-label patent classification model. By harnessing automatic feature extraction, this study outperforms the previous state-of-the-art and only requires the first 100 words of the title and abstract for precise predictions.

Both model architecture and the data the model is trained and evaluated on will make or break a model's performance. FastText is a state-of-the-art embedding model which considers n-gram chunks of words, helping to represent spelling errors or technical words which may be present in patents. There is evidence that using domain-specfic embeddings generated from patents significantly outperforms generic embeddings in patent classification tasks [6]. This study completed multi-class classification and increased the micro-average precision by 17% by expanding on previous embeddings models, namely word2vec, and considering sub-word structures when creating the embeddings. While they consider a vast corpus of documents totaling over 5 million granted patents, they only embed the first 30 words of the combined title and abstract which likely excludes important technical information contained in the patent.

While much work has been done to address pre-classification of patents, patents can belong to several categories simultaneously adding additional difficulty to correct classification. Although all US patents must be filed in English, current monolingual models are insufficient for patent retrieval in other languages [7]. This creates even larger obstacles for international patent classification, resulting in hundreds of thousands of duplicate technologies each year. This recent study utilizes a pre-trained version of XLM-R and is able to classify cross-lingual patents at the subclass level, one level deeper in the hierarchical patent classification scheme that we examine in this study. While they achieve promising results in English, German, and Chinese, their sample of languages and corpus size of documents should be expanded in the future.

3 Dataset

We start with the full public dataset USPTO Patentsview[1] consisting of over 9 million granted US patents from the last 40 years. We combine multiple available data products to produce a data set containing the patent's ID, date granted, title, abstract, and a list of all corresponding CPC classes. To constrain the required training time, we reduce our dataset using a series of cuts. First, we keep only patents granted in 2019 or later, then drop any rows where any of the data features are missing. To reduce outlier sensitivity, we remove patents with more than 8 corresponding CPC classes. This threshold was selected by visual inspection of the distribution of number of CPC classes, which is shown in Figure 1. We then randomly down-sample by a factor of $\sim$20 leaving 100,000 total samples.

For each patent, we combine the patent title and abstract to create our sample text. We remove punctuation and ensure all letters are lower-case to focus the embedding model on text semantics rather than grammatical

[1] https://patentsview.org/

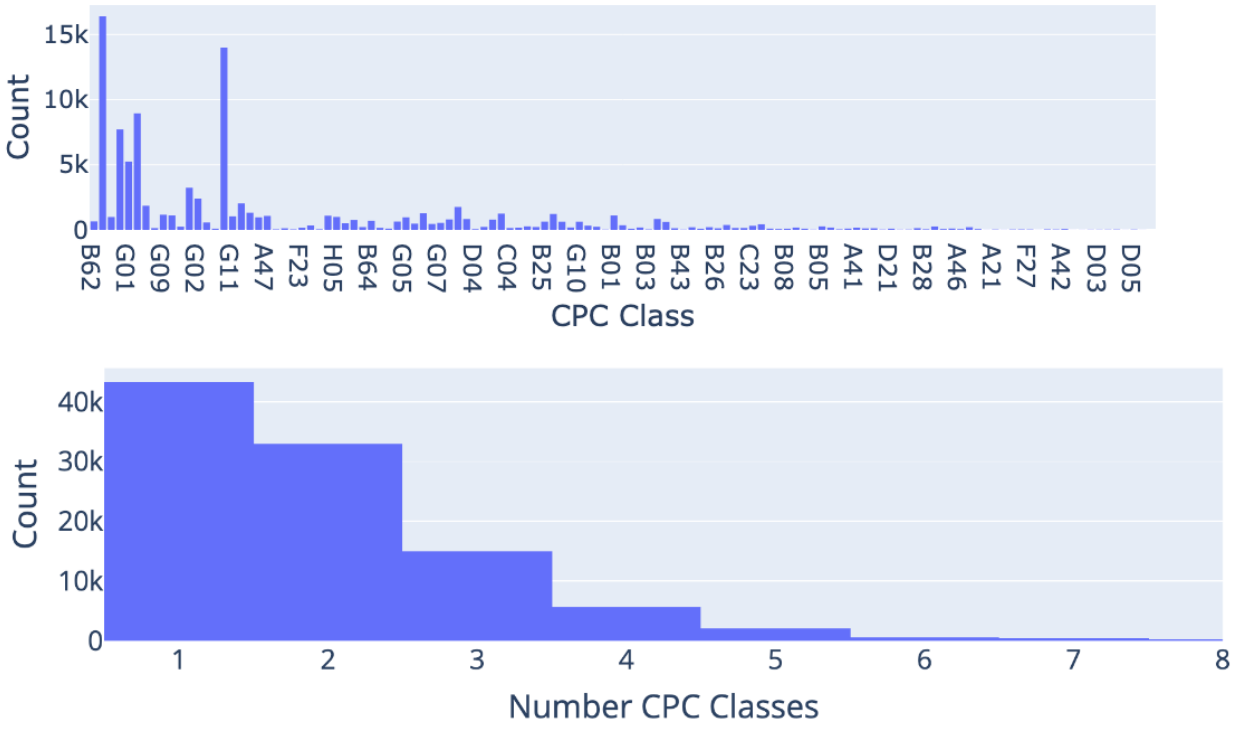

Figure 1: Top: Distribution of the CPC classes assigned to the patents in our sample. Some categories like "Electric Communication Technique" significantly outnumber others like "Bolts or Fastening Devices for Doors or Windows". Bottom: Distribution of the number of CPC classes a patent corresponds to for our sample, where we selected the cut-off to be 8 to reduce outlier sensitivity. There is an exponential decay in the number of corresponding CPC classes with the average number of CPC classes per patent being 1.94.

structure. We then create embeddings using an Augmented Sentence-BERT embedding model trained on US Patents, PatentsSBERTa[2] [8]. As previously discussed, embeddings from models trained on relevant data will outperform embeddings generated by generic text transformer models. PatentSBERTa is an augmented, fine-tuned model of the sentence transformer BERT, which is specifically trained to capture semantics in patent claims. We use this model to map the patent text to a 768 dimensional vector.

To generate the labels for supervised training, we one-hot encode the list of all CPC classes a patent belongs to. There are 130 unique CPC classes present in our dataset, although there is not an even distribution across all CPC classes as seen in the top panel of Figure 1. We explain how we account for uneven data sampling in Section 4.

In Figure 2, we show a visual representation of the 768-dimensional embeddings using the t-distributed Stochastic Neighbor Embedding (t-SNE) algorithm. This data-dimensionality reduction which calculates the probability distribution that the embeddings belong to the same group. For simplicity, we show the distribution of CPC Sections (one step above CPC class in the hierarchical patent classification scheme) for 20% of the data sample. In this low-dimensional space, the sections are not visually distinguishable.

We use 70% of the patent samples for training and 20% for model validation. We save 10% to evaluate our models' performance on unseen data.

[2]https://huggingface.co/AI-Growth-Lab/PatentSBERTa

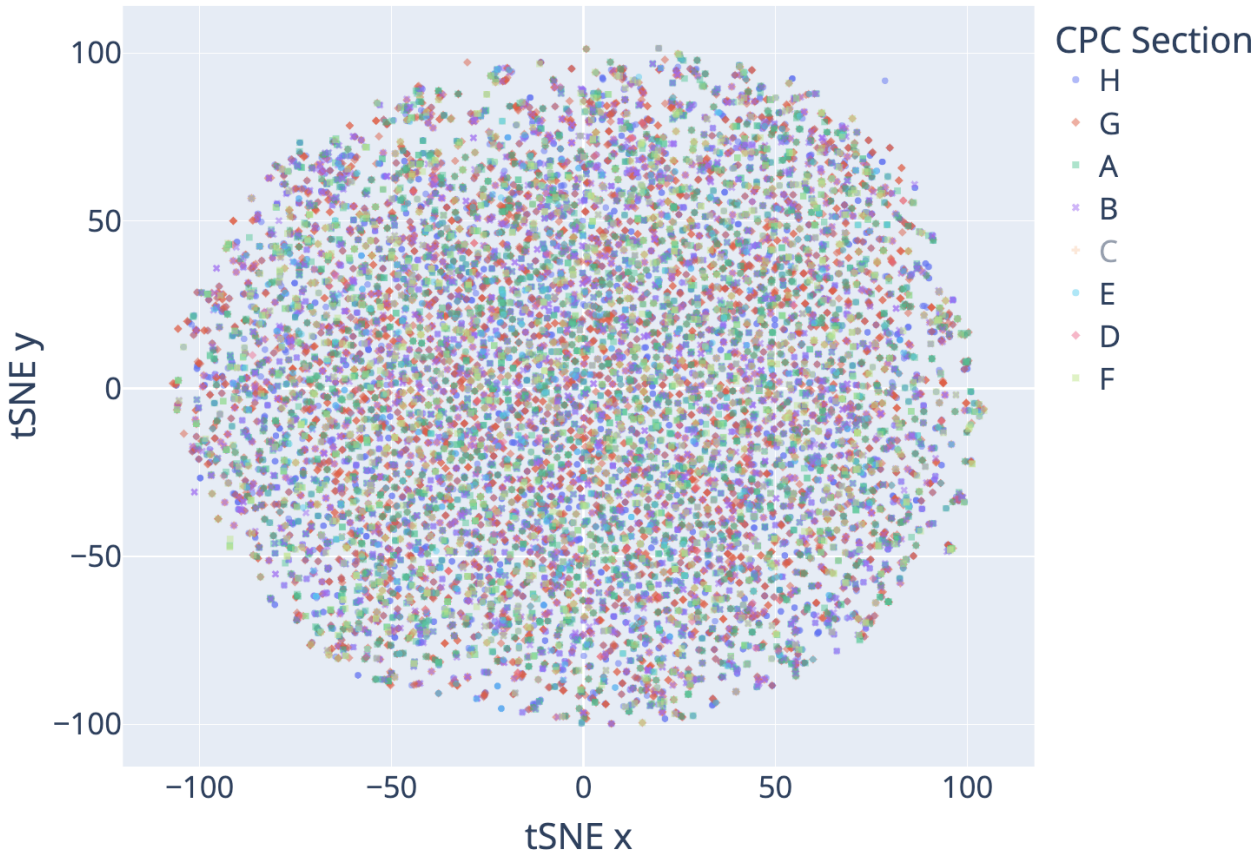

Figure 2: Visual representation of the 768-dimensional text embeddings mapped to 2 dimensions with t-SNE for 20% of the sample patents in our dataset. Each point is colored by CPC Section rather than class to increase readability. Since the dimensionality is significantly reduced from 768 to 2 dimensions, CPC sections are not visually distinguishable in this vector space.

4 Methods

We construct a multi-label classification model to map the patent text embeddings to the one-hot-encoded labels to identify membership of the 130 CPC classes examined in this study. We use Keras [9] to build, train, and evaluate our networks. When evaluating the networks, we use a threshold probability of 50% to define classification for that node, meaning the model must predict a probability of at least 50% that the patent text corresponds to the CPC class represented by the output node.

4.1 Fully Connected Network

We first examine patent classification from text embeddings using a fully connected network (FCN). FCNs are easy to understand as each node in a layer is connected to all nodes in the next layer. The 768 dimension embedding is fed into the first (input layer), then passed through the subsequent hidden layers until their CPC class is predicted at the final layer with 130 nodes. Each node in this output layer represents the probability that the patent embedding corresponds to the class represented by the node. Our model contains 12 hidden layers, for a total number of 340,730 trainable parameters. We include a hidden layer with 200 nodes each using the Rectified Linear Unit (ReLU) activation between each layer. In between each of these layers we include a Dropout layer which randomly assigns 30% of the nodes a 0 probability during each epoch of training to prevent overfitting of the data. The output layer consists of 130 nodes with a sigmoid activation function given by

$$S(x) = \frac{1}{1 + e^{-x}} \tag{1}$$

where x is the probability given by the previous node. We use a binary cross entropy function for training given by

$$\mathcal{L} = -\frac{1}{N} \sum_{i=1}^{N} y_i \cdot \log(p(y_i)) + (1 - y_i) \cdot \log(1 - p(y_i)) \tag{2}$$

The goal of the training is to minimize the log loss given by this function by adjusting the weights of the parameters (weights) connecting each node in the network. We use the Adaptive Moment Estimation (Adam) optimizer to calculate the gradient of the loss function, and we monitor the accuracy and binary-accuracy of our model during training. We train for 30 epochs over 72,000 samples, requiring 3 minutes of time on 8 input nodes. We achieve a 57.7% accuracy when comparing to our unseen test sample. We note that the accuracy metric requires that our model correctly predict every class a patent corresponds to every time. Thus we also monitor the binary accuracy metric which calculates the percentage of predictions which match the binary labels, achieving a 99.1% accuracy on unseen test data.

4.2 Fully Connected Network with Weights

Because the number of patent CPC classes in our sample is imbalanced, we train a second FCN with weights applied to the classes. This ensures that CPC classes with fewer patent samples are counted as more instances of the data, while classes containing larger numbers are counted as fewer instances. Since a handful of classes dominate our training and validation datasets, this step provides more accurate and robust training of the weights.

We use Scikit Learn to compute the relative weights of the classes based on the number of patent samples available for that class. We then train and evaluate the same network described in Section 4.1 with the class weights applied. This model also takes 3 minutes for 30 epochs on 8 input nodes to train. We find a lower accuracy and binary accuracy for our test sample with 48.2% and 98.8% respectively.

4.3 1D Convolutional Neural Network

Motivated by previous works using CNNs for patent classification, we also develop a convolutional neural network (CNN) to predict CPC classes. We feed the same text embedding into the input layer of our network. It then passes through a 1-dimensional convolution layer with 64 filters and a kernel size of 3. This layer learns patterns and features within the text embedding data by parsing the data in chunks of 3 within

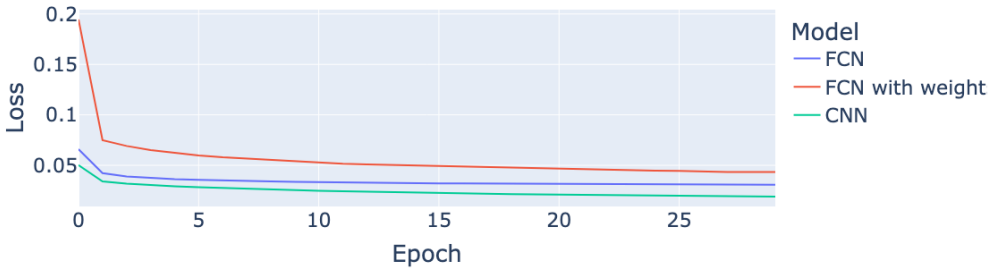

Figure 3: Loss curves for the three models described in Section 4 over 30 epochs. All three models exhibit consistent decrease in the binary cross entropy loss function indicating successful training, with the CNN producing the smallest loss as expected.

	FCN	**Weighted FCN**	**CNN**
Avg. Precision	0.48	0.46	0.52
Avg. Recall	0.22	0.31	0.28
Avg. F1	0.28	0.33	0.34

Table 1: Macro-average summary of the three models we examined: fully connected network (FCN), weighted FCN, and 1D convolutional neural network (CNN). In all three metrics, the 1D CNN outperforms the other two models.

the embedding vector. The kernel can be thought of as a sliding window which views three components of the embedding at a time, updates the weights of the model, the moves over one index to examine the next consecutive group of three. This means that each entry in our text embedding (except for the first two and the last two) are examined 3 times within the context of its surrounding neighbors. The network repeats this process 64 times for the 64 different filters, where the sliding-window kernel learns the best weights for each of the filters. Thus the output of this layer is 64 different vectors of length 766, since the kernel only passes over the first and last entry in the text embedding once. We use the ReLU activation function and connect this layer to an identical convolutional layer. Using two convolutional layers allows the model to learn hierarchical features in the data and automatically extract important features. We then implement a Dropout layer with a frequency of 50% to reduce overfitting before feeding these results into a max pooling layer. We use a pool size of two meaning that for each consecutive pair of entries from the previous layer, the larger of the two is kept. This layer increases the models resiliency to data variances like spelling or grammatical order and will focus on stronger features represented by the embeddings like technical vocabulary which will aide with classification. Finally, we connect the convolutional network to a single fully connected layer with 100 nodes and ReLU activation before passing the weights to output layer with a sigmoid activation function. We train for 30 epochs on 8 input nodes requiring 30 minutes. The CNN produces an accuracy of 56.6% and a binary accuracy of 99.0% respectively.

In Figure 3, we show the loss curves for all three models for training over 30 epochs. All three models demonstrate clear learning in minimizing the binary cross entropy function.

5 Results/Discussion

During training, we used the default hyperparameter settings. This included a batch size of 128 (for 563 iterations per epoch) and a learning rate of of 0.001. Given a steady decrease in the loss functions for each of the three models, we did not adjust these. In all three of our models, we include Dropout layers to reduce the dependence of the predictions on any single node, effectively mitigating overfitting of our data.

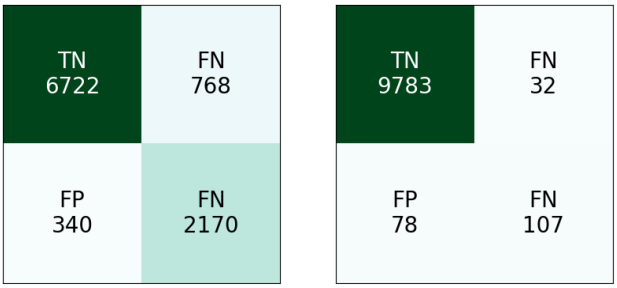

Figure 4: Confusion matrix heatmaps displaying CNN model performance. The left matrix shows the performance for the top-performing class, and the right matrix shows the typical class performance. In either case, the model excels at predicting true negatives but struggles to identify true positive labels.

Our model assigns labels based on the CPC class of the text embedding, and the performance of the model varies from class to class. We therefore use macro-average metrics when evaluating the performance of the model. For each class, we calculate the precision, recall, and F1 score, then average these results across all classes. Each model's performance is summarized in Table 1. The weighted FCN has a lower precision than the first FCN, but its recall and F1 are notably higher. This indicates a more robust performance because the model balances both precision (identification of true positives against all positive predictions) and recall (identification of true positives against all positive samples). The convolutional neural network outperforms the other two in all three metrics, suggesting the best overall model performance. This likely occurs because the CNN is able to extract relevant features through convolutional and max pooling layers, enabling it to give even higher priority to technical vocabulary words rather than filler words like 'the'.

Overall our model does not perform as well as expected, as these metrics are dominated by the correctly identifying which classes a patent doesn't belong to, instead of those that it does. For example, about 8% of the patents in our test sample receive no clear class prediction. Results for the other patents can be viewed in Figure 4. We present two confusion matrices showing the CNN performance for the best-performing class, G06, and the typical performance of our model with the A61 class as an example. The top left square is the number of true negatives (TNs) and the bottom right square is the number of true positives (TPs) which should be maximized if the model performs well. The top right is the number of false positives (FPs) and the bottom left is the number of false negatives (FNs), which should both be minimized. In both cases the model succeeds at identifying TNs. In the left panel, it identifies TPs reasonably well, but also predicts about 50% as many FPs and FNs as it identifies TPs. In the panel on the right, which is representative of most patent classes, the model isn't able to identify any more TPs than it is FPs or FNs. This likely occurs because of our labeling method for classifications, in which we one-hot-encode a 130 dimensional vector. As seen in Figure 1, there is an exponential decrease in the number of CPC classes a patent is assigned to, meaning that our labels are sparse matrices with few positive entries. Thus each network will give significant weight to predicting 0 for all classes, and struggle to create positive predictions, true or not.

To analyze this effect more, we examined the dependence of the F1 score on the number of patents corresponding to a class. We expect that more patents available for training and validation would yield a higher F1 score for that class. Figure 5 shows the F1 score per class for each of the three models as a function of the number of total patents in the sample. There is not a substantial difference between the models, and there is no apparent trend between the number of patents available during training and validation with the overall model performance. This further suggests that the dataset itself is not causing sub-par model performance, but rather how we have chosen to label the CPC classes.

6 Conclusions/Future Work

In this work we trained and evaluated three networks to predict patent categorization from their title and abstract. We utilized PatentSBERTa embeddings to find the relationship between patent text and the corresponding CPC class(es). We map 768 dimensional text embeddings to 130 dimensional class predictions

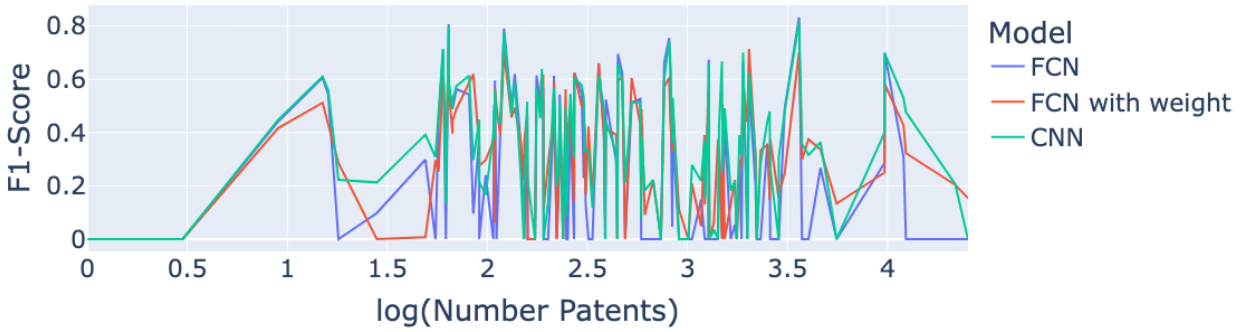

Figure 5: F1 scores for each class as a function of the number of patents assigned to that class for our total patent sample. All three models perform similarly, and there is no clear trend between the number of available patents and class performance.

using an FCN, an FCN with class weights and a 1D CNN. Our models excel in identifying true negative classification, but have difficulty predicting true positives due to the sparsity of one-hot-encoded CPC class labels. Of the three models, the CNN outperforms the fully connected networks in accuracy, recall, and F1 score. This is consistent with expectations given that convolutional networks learn key features and patterns, and our model includes a max pooling layer which will reduce the likelihood of predicting negative values too frequently.

There are several possibilities to improve model performance which are beyond the scope of this project. Future works should utilize more sophisticated labeling techniques to reduce the sparsity of the one-hot-encoded matrics we use here, generating fewer FN and more TP predictions. For example, our labels do not account for the hierarchical nature of the CPC classification system, where multiple classes belong under the same Section. Thus our labels encode no information that patents belonging to classes B21 and B22 will contain similar information. This feature can strengthen class weights, and also improve representation of classes with fewer patents, overall providing a more robust model performance and more strongly distinguish between similar classes. We propose encoding the labels themselves through text embeddings or a similar method such that the CPC Section and class number are considered in relation to the others during training and inference. Future works might also consider increasing model complexity, such as including more nodes or varying the number of nodes per layer. For the convolutional network, one could also experiment with the dimensionality of the convolution by reshaping the embedding inputs. This method would help relate earlier parts of the patent text like the title to later parts where relevant technical vocabulary might appear. Finally, future works should look to incorporate more data, especially for patents from underrepresented classes, which will improve the precision and recall of less common classes.

References

[1] US Patent and Trademark Office, www.uspto.gov

[2] Shalaby, W., and Zadrozny, W. (2019). Patent retrieval: a literature review. Knowledge and Information Systems, 1–30.

[3] Haghighian Roudsari, A., Afshar, J., Lee, W. et al. PatentNet: multi-label classification of patent documents using deep learning based language understanding. Scientometrics 127, 207–231 (2022). https://doi.org/10.1007/s11192-021-04179-4

[4] Devlin J., Chang M., Lee K., and Toutanova K., BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding, North American Chapter of the Association for Computational Linguistics, 2019, https://api.semanticscholar.org/CorpusID:52967399.

[5] Li, S., Hu, J., Cui, Y. et al. DeepPatent: patent classification with convolutional neural networks and word embedding. Scientometrics 117, 721–744 (2018). https://doi.org/10.1007/s11192-018-2905-5

[6] Risch, J. and Krestel, R. (2019), "Domain-specific word embeddings for patent classification", Data Technologies and Applications, Vol. 53 No. 1, pp. 108-122. https://doi.org/10.1108/DTA-01-2019-0002

[7] Lu, Y., Chen, L., Tong, X. et al. Research on cross-lingual multi-label patent classification based on pre-trained model. Scientometrics 129, 3067–3087 (2024). https://doi.org/10.1007/s11192-024-05024-0

[8] Bekamiri, Hamid and Hain, Daniel S and Jurowetzki, Roman, PatentSBERTa: A Deep NLP based Hybrid Model for Patent Distance and Classification using Augmented SBERT, arXiv preprint arXiv:2103.11933, 2021.

[9] Chollet, François and others, Keras, 2015, `https://keras.io`