Artificial Intelligence and Law
https://doi.org/10.1007/s10506-025-09483-5

ORIGINAL RESEARCH

Automated neural patent landscaping in the small data regime using citations and CPC codes

Tisa Islam Erana[1] · **Mark A. Finlayson**[2]

Received: 13 April 2024 / Accepted: 2 September 2025

Abstract

Patent landscaping is the process of identifying all patents related to a particular technological area, and is important for assessing various aspects of the intellectual property context. Traditionally, constructing patent landscapes is intensely laborious and expensive, and the rapid expansion of patenting activity in recent decades has driven an increasing need for efficient and effective automated patent landscaping approaches. In particular, it is critical that we be able to construct patent landscapes using a minimal number of labeled examples, as labeling patents for a narrow technology area requires highly specialized (and hence expensive) technical knowledge. We present an automated neural patent landscaping system that demonstrates significantly improved performance on difficult examples (0.69 F_1 on 'hard' examples, versus 0.6 for previously reported systems), and also significant improvements with much less training data (overall 0.75 F_1 on as few as 24 examples). Furthermore, in evaluating such automated landscaping systems, acquiring good data is challenge; we demonstrate a higher-quality training data generation procedure by merging (Abood and Feltenberger Artif Intell Law 26:103–125 2018) "seed/anti-seed" approach with active learning to collect difficult labeled examples near the decision boundary. Using this procedure we created a new dataset of labeled AI patents for training and testing. As in prior work we compare our approach with a number of baseline systems, and we release our code and data for others to build upon "(Code and data may be downloaded from https://doi.org/10.34703/gzx1-9v95/QDLKVW Code and data are released under the Creative Commons NC-BY 4.0 license at https://creativecommons.org/licenses/by-nc/4.0/)".

Keywords Patent landscaping · Active learning · Deep neural network · BERT for patents · Citation network · Classification code

Extended author information available on the last page of the article

Published online: 04 October 2025

1 Introduction

In its simplest form, patent landscaping is the process of identifying all patents that are related to a particular technology or technology area. Patent landscapes are useful for a number of activities: it is important for assessing the coverage, value, or context of particular pieces of intellectual property, or for understanding the direction, speed, or concentration of innovation in a particular industry (Hunt et al. 2007). For example, companies create patent landscapes to evaluate the risks posed by competitors in a particular technology space, or to decide whether and how much to invest in pursuing particular innovations. Patent offices and economic monitoring organizations use patent landscapes to evaluate how a particular technology is affecting or might affect the economy, for example, how much economic investment is underway in a technology, how much economic value has been generated, or how many industries or companies are supported by a particular technology. Governments, in turn, can use that information to implement technology policies, for example, deciding whether to steer investment or tax incentives to companies working in particular areas (e.g., AI or green technologies). While the simplest form of patent landscaping merely identifies which patents are related to a particular area, other more sophisticated forms of patent landscaping can seek to identify how different subareas of a technology area are related, which companies or inventor groups are the most prolific, what regions are involved, or what specific types of innovations are the focus of current development.

Patent landscaping must be clearly differentiated from patent *classification*. Patent classification refers to the assignment of a patent application or patent to one or more standardized technology classes, such classes those found in the Cooperative Patent Classification (CPC) hierarchy used by the US Patent & Trademark Office (USPTO) and the European Patent Office (EPO). Patent classification is an early (now usually automated) step in patent examination—the process of a patent office evaluating a patent application— and is used to route a patent application to patent examiners with the correct expertise. Patent landscaping differs from patent classification because a standardized patent class scheme may not contain classes or distinctions relevant to a particular patent landscaping question or need.

Patent landscaping, even its simplest form, can be intensely laborious and expensive for at least two reasons. First, the technical expertise needed to evaluate whether a patent should be included in or excluded from a landscape is often quite specialized, and the experts possessing that knowledge are rare and their time is expensive. Consider, for example, a company that develops airbags in vehicles, and needs to know the patent landscape related to *sensor-integrated variable and adaptive ventilation, opening via pressure regulation and resulting in enhanced occupant protection*. There may only be a handful of people in the world with this expertise. Second, the number of patents and patent applications is rapidly increasing and shows no signs of abating: the number of patents issued per year by the USPTO alone has doubled since 2002, from a total of 177,312 to 361,435 patents (USPTO 2022). Indeed, after 228 years of patenting activity, the USPTO issued it's ten-millionth patent in 2018. The eleven-millionth patent was issued a mere 3 years later. Therefore, automated approaches to patent landscaping are sorely needed.

As will be reviewed in Section 2, at least two research systems have been developed to tackle automated patent landscaping (c.f. Abood and Feltenberger, 2018, Choi et al., 2019). As is the case with much recent work in NLP, these are deep neural systems, and they presumably can be adapted to different technology areas given new labeled data. While effective in many ways, these prior systems have two shortcomings. First, due to challenges of assembling good labeled data, both of these approaches were trained only on "easy" examples found via keyword search or patent family expansion; as a consequence, they do not work well on "hard" examples near the landscape boundary. Second, the models were trained on several thousand positive examples and tens of thousands of negative examples. While this generates good performance (at least on "easy" examples), for many technology areas this amount of labeled data would in many cases be prohibitive, in that it would be extremely expensive to obtain via manual labeling, and many technology areas (especially if quite specialized) might not even contain that many positive examples. Further, such a training setup also presents data balance issues.

In this work we attempt to address these two issues (data quality and amount of data needed). First, starting from the "seed/anti-seed" approach of Abood and Feltenberger (2018) which generates "easy" examples, we leverage active learning to collect a set of labeled "hard" examples (positive and negative) near an estimated landscape boundary. This allows us to build a much higher quality dataset for training and testing and evaluate prior architectures as to their performance on these more difficult cases. Second, we investigate the incorporation of additional features into the neural classifiers—in particular citation networks (which have not been previously used in conjunction with deep neural methods), and CPC code embeddings (as was done in Pujari et al. 2022)—which significantly improve performance when using a small number of training examples. We show an overall performance of 0.75 F_1 on as few as 24 examples, which is comparable to prior work trained on hundreds of times as much data.

The paper is organized as follows. We first discuss the prior work on automated patent landscaping (§2). Next we present our data collection approach, reviewing our annotation tool and the dataset we generated using that tool (§3). We then describe our architecture, which builds on a variety of prior work but combines them in new ways (§4). We review our experiment setup and results next (§5). Finally, we conclude with a brief discussion (§6), limitations & future work (§7) and a list of our contributions (§8).

2 Related work

2.1 Patent classification

As noted above, patent classification (as opposed to patent landscaping) is the process of assigning a patent classification code from an established code list (such as the CPC code hierarchy) to aid patent examination and search. An example of a section of the CPC code hierarchy is shown in Fig. 1. Patent classification is related to, but must be kept distinct from, patent landscaping. Traditional approaches to patent

```
{CPC Section: A - HUMAN NECESSITIES
  └ CPC Class: A01 - AGRICULTURE; FORESTRY; ANIMAL HUSBANDRY; HUNTING; TRAPPING; FISHING
    └ CPC Subclass: A01B - SOIL WORKING IN AGRICULTURE OR FORESTRY; PARTS, DETAILS, OR
      │                   ACCESSORIES OF AGRICULTURAL MACHINES OR IMPLEMENTS, IN GENERAL
      └ CPC Group: A01B1/00 - Hand tools
        └ Main Group: A01B1/02 - Spades; Shovels
          └ Subgroup: A01B1/024 - Foot protectors attached to the blade
```

Fig. 1 A section of the CPC code hierarchy showing CPC class, subclass, main group and subgroups

classification involve converting text features of a patent—such as abstract, claim or title—into feature vectors, and then passing these features through a supervised classification model, such as a Support Vector Machine or a K-Nearest Neighbour model (Yun and Geum 2020; Seneviratne et al. 2015). Other work has used the bibliographic metadata as the primary feature for labelling patent with classification codes. For example, Li et al. (2007) used the CPC codes of the patents cited by a patent—both single hop (direct citation) and multi-hop (citation networks)—to create features for classification in a featurized supervised machine learning model. More recent approaches use deep neural models for patent classification. Since patents contain different types of text data like titles, abstracts, claims etc. embedding techniques like Word2Vec (Mikolov et al. 2013) and language models like BERT (Devlin et al. 2018) have been used to great effect. For example, Lee and Hsiang (2020) fine-tuned BERT using a patent classification task, resulting in a model they call PatentBERT. DeepPatent (Li et al. 2018) used a Convolutional Neural Network and word embeddings of the patent text to classify patents into CPC codes.

2.2 Automated patent landscaping

There have been several lines of effort that directly addressed the problem of automated patent landscaping. Abood and Feltenberger (2018) presented a deep learning based method to select patents relevant to a particular topic from a broader set of candidate patents. Since there was no benchmark dataset available for training patent landscaping models, they developed a general method to build training data for any particular topic. Their method begins with taking a small number ($\sim$1000) of "seed" patents curated by experts that are the positive examples of a technology. From these they created a "Level 1" (L1) set by finding patents that share CPC codes with or are cited by the seed patents. A "Level 2" (L2) set is created by including patents that share a "family" relationship with those in L1 (an expression of patent priority or patent continuation). Any patent outside of L2 is considered a negative example (or an "anti-seed") for the topic. They then trained a deep neural network using seeds and anti-seeds as positive and negative training examples, and then applied the classifier to the patents in L2 to find patents that should be included in the landscape (which already includes the seeds). Their model architecture was a wide and deep Long-Short Term Memory (LSTM; Hochreiter and Schmidhuber, 1997), and they used Word2Vec (Mikolov et al. 2013) embeddings of words in the abstract as the sequential input input to the LSTM network. The metadata of the patents—the citations and the CPC codes—were represented by 1-hot encoding vectors. The metadata vectors and the output of the LSTM handling the abstract were concatenated and

used as input to a multi-layer perceptron (MLP) network to perform a binary classification. They built landscaping models for four topics: *browser*, *operating system*, *video codec* and *machine learning*. They reported F_1 scores above 0.95 for each of these topics. Despite this impressive performance, the work has several shortcomings. First, the training data generated by the seed/anti-seed method does not provide positive or negative examples near the landscape boundary (i.e., they only include "easy" examples). In our experiments we obtain patents near the landscape boundary via human labeling and show that Abood and Feltenberger (2018) performance is quite low on these examples. A smaller problem is that if a patent that does not share a CPC code, citation, or family link with one of the seed patents, it will never be included in the landscape (because the approach *filters* patents from the L1 and L2 sets), which is potentially problematic for broad landscapes covering many CPC codes, or landscapes that include many patents (where citation networks are unlikely to be exhaustive). Abood and Feltenberger (2018) work was continued by Giczy et al. (2021), where they removed the CPC codes and added the claim texts as features. They did not measure the model performance; but rather they used it to create a landscape of AI patents.

Choi et al. (2019) is the second line of effort, which also uses deep learning. They used word2vec embedding of the patent abstracts as inputs to a modified transformer architecture comprising both multi-head self-attention and scaled dot product attention. They also experimented with diffusion graph embedding techniques for representing IPC, CPC, and USPC classification codes. They used four topics analyzed in the Korea Intellectual Property Strategy Agency (KISTA) Patent trends reports (KISTA 2023), which provided positive examples for training. They generated negative examples by repeating the keyword searches used by the human experts when generating the original landscapes, and using patents returned by that search but not included in the landscape. Because negative examples vastly outnumber positive examples, they performed under-sampling of negative examples using CPC codes of the target patents. They reported F_1 scores ranging from 0.62 to 0.89. They also evaluated (Abood and Feltenberger 2018) approach on their data, and in all cases showed better performance. They also compared with a baseline model based on PatentBERT which performed similarly to their model.

Antonin and Cyril (2023) also investigated neural approaches to landscaping, using a modified version of Abood & Feltenberger's data collection procedure. They first defined rules to identify a set of manually pre-identified candidates: these rules leveraged features such as technological classes, keywords, and patent similarity. These rules were then used to define the initial seed set, which was manually reviewed. Then they performed seed expansion as described in Abood & Feltenberger. They experimented with different neural models, including multi-layer perceptron (MLP), convolutional neural nets (CNN), and a transformer architecture. They reported F_1 scores ranging from 0.65 to 0.97, although for nearly all experiments their techniques do not outperform (Abood and Feltenberger 2018) approach.

Finally, Pujari et al. (2022) developed systems for exposing structure *inside* of a landscape (which they called "patent-landscape-oriented target classification"). In their task, they started with a technology area and had experts provide a set of categories that represented important features or aspects of the patents in the landscape,

and labeled a part of landscape with these categories (which was a multi-label classification task) They experimented on three datasets—focused on the topics of *Injection Values*, *Ritonavir*, and *Atazanavir*—and used the same classification architecture they proposed in previous work on patent classification (Pujari et al. 2021). This architecture comprised a Transformer-based Multi-task Model (TMM) which took as input SciBERT (Beltagy et al. 2019) embeddings for title, abstract, claims, and description. They also experimented with different ways of computing graph embeddings of the CPC labels, which were used as an additional input. They achieved F_1 s of 0.68 to 0.84.

3 Data

As discussed above, the data collection method developed by Abood and Feltenberger (2018), and used in follow-on papers by Giczy et al. (2021) and Antonin and Cyril (2023), starts from a set of manually annotated positive examples (called "seeds", of which they used around 1,000 seeds for each technology domain). They are able to rapidly generate a large number of negative examples ("anti-seeds") by sampling from patents which do not share a CPC code with a seed, do not have a patent family relationship with the seeds, and are not cited by the seeds or their family members. While this method is highly scalable, we had questions about whether these examples were actually modeling the landscape boundary well. It seemed plausible that, especially for a large technology domain, the seeds would not necessarily be near the boundary, and anti-seeds would almost certainly be far from it.

To investigate this hypothesis, we sought to collect manually annotated positive and negative examples that were near the landscape boundary, to see whether (Abood and Feltenberger 2018) technique maintained its performance in this region. The idea was to start from seed/anti-seeds collected in the manner described by Abood and Feltenberger (2018), but then use these to identify examples nearer to the landscape boundary and manually annotate them. We used active learning and an annotation tool of our own design to accomplish this, as described later in this section.

To begin, we chose the technology domain of *Artificial Intelligence* (AI) as the defining topic of the landscape, and worked exclusively on patents downloadable electronically from USPTO's *PatentsView* (USPTO 2019) data repository. This set numbered more than 2 million patents as of 2021. We further obtained 2,020 seed positive examples of patents in the AI space from the USPTO itself, as reported by Giczy et al. (2021). These examples were annotated by patent examiners with expertise in AI employed by the USPTO. The USPTO also had performed (Abood and Feltenberger 2018) L1/L2 expansion to generate a set of anti-seeds / negative examples, from which they randomly sampled 56,093 anti-seeds which they provided to us.

3.1 Annotation using active learning

To enable the collection of data for active learning, we first developed an annotation tool that we call *Patentify*. Patentify has a front-end user interface for labeling patents

(shown in Fig. 2), and runs an active-learning-based Support Vector Machine (SVM; Boser et al., 1992) on the back-end to identify patents near the decision boundary for labeling. Active learning is a semi-automatic machine learning technique which interacts with the user in a feedback loop and results in higher quality and more informative labeled examples (Settles 2012). We chose SVM over other complex neural models for the annotation tool because it is easy to implement, is not compute intensive, and the results of a linear kernel classification are easily interpreted. The SVM model on the backend of *Patentify* uses as features the *tf-idf* counts of all the words (except stop words) in the title and abstract text of patents. We selected the "linear" kernel and value of C as 2. The model was then run over all 2 million patents that are extracted from the PatentsView data, and saved into *Patentify* 's database. In order to filter the examples near the decision boundary, we used uncertainty sampling (Lewis and Gale 1994) to prioritize the patents that the SVM model is most uncertain about. The active learning loop begins with the backend system randomly sampling a set of documents from the database. These samples are then fed to the SVM for prediction, and the system creates a ranked list based on the prediction probability score of the SVM with the most uncertain samples placed at the top. When the users log into *Patentify*, these decision boundary example patents are presented to them for labeling. Once a set of new patents are annotated, these decision boundary examples are incorporated to the training dataset to retrain the SVM model, thus improving its performance with each iteration. The admin user also has the option of assigning patents to specific annotators. This was implemented for the purpose of measuring inter-annotator agreement.

In order to setup the tool for labeling and training the back-end machine learning model, we initially built a class-balanced dataset of 476 randomly sampled seeds and anti-seeds from the USPTO-provided data, which we used to train an initial, baseline SVM classification model. In the first active learning pass, two annotators in our lab annotated a set of 462 examples near the decision boundary (equally split between positive and negative). Of these, 50 examples were annotated by both to estimate pre-

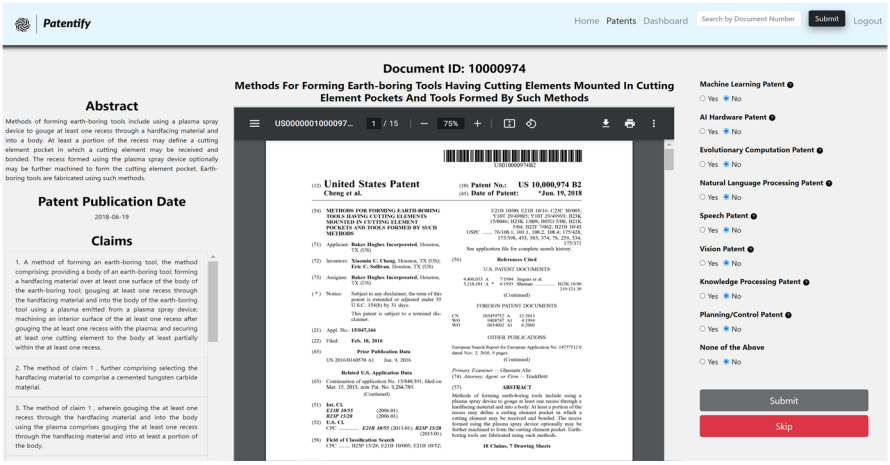

Fig. 2 Screenshot of the *Patentify* Annotation Tool

liminary inter-annotator agreements, while the remaining 412 examples were divided between the two annotators. These annotators were trained by the first author. Their training revealed problems in the training procedure which we addressed in the next stage, by engaging a trained patent examiner. Nevertheless, our manual review of this initial set of annotations demonstrated that the data was good enough to provide a "rough cut" of annotated data to seed the model. Once collected, these annotated examples were added to the original set of seeds and anti-seeds to train the second version of the model.

In the next stage of the annotation, we expanded the annotator group to 14 annotators. Our annotators were graduate and undergraduate students in our laboratory with at least 2 years of research experience in AI. One of our collaborators, a USPTO patent examiner, trained the annotators on how to read patents and determine whether they should be labeled as AI. According to the trainer, a patent should be considered AI if the problem the inventors are trying to solve, or the solution itself, substantively relates to AI. "Substantive" was defined by examples provided by the patent examiner. Annotators were asked to skim the entire document so that they had a comprehensive understanding of the invention. In addition to the abstract, the front-page components, and the figures, the patent examiner recommended reading the background and summary carefully, as well as skimming the detailed description to determine whether the technology of the invention substantively involves AI. The patent expert cautioned that inventors might use their own terminology and advised annotators not to label a patent as AI solely based on certain keywords or because only a small portion of the text mentions an AI component. Each annotator was asked to annotate approximately 100 patents (sampled by *Patentify*) as whether or not they involved AI. After every 10 labeled examples (positive or negative) collected across all annotators, *Patentify* retrained the SVM model and reranked the most informative patents for labeling. Every annotator was presented with patents from the updated ranked list, ensuring that all annotators worked with the most informative examples available at that stage. Individually annotators spent approximately four hours each to do their annotations, for a total annotation effort of around fifty hours.

In order to measure the inter-annotator agreement, we selected two annotators from our laboratory and each was assigned another set of 100 patents to perform further annotation, that were previously done by other annotators. Across these two sets of 100 patents, we measured an inter-annotator of 0.806 Cohen's κ, which is considered "almost perfect" (Landis and Koch 1977) or "excellent" (Fleiss 1981) agreement. We did not measure any inter-annotator measures for the "easy" (seed or anti-seed) patents, as these were originally found by keyword search. The measured inter-annotator agreement on the "hard" examples indicates that identifying "substantive" use of AI in a patent was relatively easy for the annotators.

3.2 Dataset

We collected 1,149 annotated examples comprising 395 positive examples and 754 negative examples. We then separated our data into four categories: positive vs. negative examples crossed hard vs. easy examples, as shown in Table 1.

Table 1 Data collected for each type of example (hard vs. easy × positive vs. negative)

Type	#	How collected
Easy+ (seeds)	2,020	Manual anno. by USPTO exam.
Easy- (anti-seeds)	56,093	Samples outside of L1/L2
Hard+	395	Manual anno. by AI students
Hard-	754	Manual anno. by AI students

Table 2 Summary of the datasets

Dataset	Hard+	Hard-	Easy+	Easy-	Total
All Data / Unbalanced	395	754	2,020	56,093	59,262
Full Balanced	395	395	395	395	1,580
Full Holdout Test	0	359	1,625	55,698	57,682

Underlined sets are randomly sampled and vary when generating different versions to investigate statistical variation

These data allowed us to construct several different types of training and testing datasets, as shown in Table 2. First, we constructed an *All Data / Unbalanced* set with 59,262 examples (2,415 positive, 56,847 negatives), comprising all the data. Second, we constructed a *Full Balanced* dataset comprising all 395 available hard positive examples (the smallest category), and an equal number of examples randomly sampled from the other categories (hard negatives, easy positives, easy negatives), resulting in 1,580 total examples (790 positive, 790 negative). Because the random samples differed between experiments, we obtained multiple versions of the Full Balanced dataset and the corresponding Full Holdout Test set. In each experiment, we conducted 5-fold cross validation, and we report statistical variation in terms of mean and standard deviation across folds shown in Table 4.

Third, we constructed a *Full Holdout Test* dataset from the examples not included in the Full Balanced set, comprising 57,682 examples (359 'hard' negatives, 1,625 'easy' positives, and 55,698 'easy' negatives). Again, when we generated a different Full Balanced dataset, this necessarily generated a complementary Full Holdout Test dataset. Note that the Full Holdout Test set contains hard negatives but no hard positives. This is because we incorporated all 395 available hard positives into the Full Balanced dataset in order to maximize their use for training, as this category was most limited.

4 Patent landscaping methods

Our approach begins from the architecture described by Abood and Feltenberger (2018), and we show how we were able to achieve higher performance both using our higher quality training data and different embedding strategies. Patents are structured documents and contain several different types of information that can be used to inform landscaping, with the two major types of information being text and metadata. Text information comprises the actual language of the patent and includes fields such as the *title*, *abstract*, *claims*, and *description* (which can be further subdivided in some cases). Of the text information, the abstract provides a brief summary of the

invention, the description provides expanded explanation and context for the invention, and the claims describe the exact legal scope of the invention. Metadata includes classification codes, citations, names of inventor, assignee and applicant, filing and publication dates, and so forth. The classification codes and citations provide the most information for the purposes of basic landscaping.

dummy

4.1 Our neural architecture

Our model builds on the approaches of two prior works. Abood and Feltenberger (2018) model used the abstract, citations, and CPC codes. Each word in the abstract was encoded with word2vec and then fed into an LSTM followed by a dense MLP. Citations were encoded as a 1-hot vector then fed through two dense MLP layers. CPC codes were also encoded as a 1-hot vector and fed through two dense MLP layers. All three streams were fed into a single dense MLP followed by a dense binary classification layer. Giczy et al. (2021) refined (Abood and Feltenberger 2018) implementation by removing the CPC codes and adding the claim text.

In our neural architecture (Fig. 3), we have five input streams: three for text, one for citations, and one for CPC codes (they are numbered for ease of reference in the figure). As shown, text is encoded using embeddings (second layer of boxes from the bottom) and then fed into LSTM layers (third layer). Citation information and the average CPC code embedding vectors (second layer) are fed into single dense layers (third layer). All streams end with a single dense layer (fourth layer), which then combine in single dense layer (fifth layer) before passing into a binary classification layer (sixth layer).

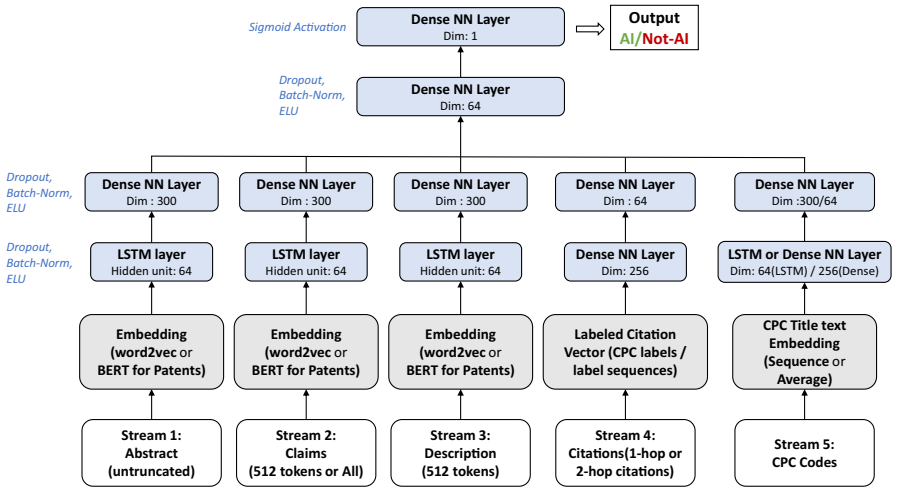

Fig. 3 Neural Architecture. We experimented with a number of variants. In particular, model variants included or excluded various streams (numbered 1–5) with various settings, as described in Table 3

Textual Information (Streams 1–3) For text we experimented with using the abstract, claims and description, encoded with word2Vec (Mikolov et al. 2013) or BERT for Patents (Rob Srebrovic 2020). BERT for Patents was pre-trained on 100 million patents and patent-related documents, with an input width of 512 tokens. It also provides a patent-specific tokenizer. Although many new embedding techniques have emerged since BERT, we chose the BERT for Patents model because it was specifically pre-trained on millions of patent documents. The domain-specific training allows this model to capture the linguistic pattern of legal language and specific terminologies of patents, thus making it more suitable for our case than a general-purpose language model. We extracted the contextualized token embeddings for abstracts and claims from the second to last encoder layer. While 512 tokens is too small for most claims sections, one can chunk the text into 512-token-sized pieces and pass them through the model one at a time to create an embedding vector longer than 512 that encodes an arbitrarily long text. When using the BERT for Patents on abstracts and claims sections, we experimented with using a single 512 token chunk (single pass) as well as using all of the text (multiple passes). We only used a single pass for descriptions because of their length. Note that Patents are public documents, part of the public record, and contain the names and addresses of inventors and assignees (which is personally identifiable information). We did not use this information in training the models.

Citation Networks (Stream 4) As shown by Li et al. (2007), citation information can be used in several ways. First, one can examine the direct outward citations of patents or patent pre-grant publications (PGPubs); this is called the *direct citation* approach. Second, one can collect further citations by following citations of citations, up to a certain number of hops (*citation network* approach). We experimented with both (stream 4). Direct citations is a special case of the citation network approach, with hops is limited to 1. Abood and Feltenberger (2018) used the direct citation (1-hop) approach, but encoded citations in a 1-hot vector, which is problematic because of the sparseness. In our experiments with Abood and Feltenberger (2018) architecture, we found that removing the 1-hot citation information actually improved performance. In our approach, we encoded 1-hop citations by representing each document as a vector of counts of CPC codes at the subclass level. Each element of the input vector to the first dense layer thus contains the number of cited documents belonging to a particular CPC subgroup code. We also experimented with using CPC codes gathered from citations two hops away from the target patent (namely, a 2-hop citation network). These were encoded as a vector of counts of two-code sequences; i.e., if the the first patent in a two-hop citation chain had code `A01B`, and the second patent had code `E05D`, then the sequence `A01B-E05D` would be incremented by 1.

CPC Embeddings (Stream 5) We embedded tokens in the concatenated CPC titles using BERT for Patents. We tried two approaches. First, we fed the embedding for each token, for each code in sequence, one at a time into an LSTM layer. Second, we computed an average embedding for the set of codes for a patent by averaging the embeddings of the tokens, place-wise, meaning the embedding vectors for token 1 of

all codes was averaged together to produce an average token 1 embedding, the same for token 2, and so forth. This vector was then fed into a single dense layer.

4.2 Models tested

Figure 3 illustrates the design choices in our architecture. We experimented with different combinations, listed in Table 3. To compare with our models on the same data, we obtained or created implementations of Abood and Feltenberger (2018); Giczy et al. (2021) and Choi et al. (2019) architectures. We refer to these models as `A&F`, `A&F/USPTO`, and `Choi` respectively. We also created five baseline SVM models (RBF kernel). The first baseline was a regular Bag-of-Words approach over abstracts and claims with the words represented as *tf-idf* vectors (Zhang et al. 2011) (vocabulary size of 49,639 [abstracts] and 77,989 [claims]; cut-off value of 1.0). We call this baseline `SVM/tfidf`. The second baseline used word2vec vector embeddings of the abstracts and claims, on which we performed principal component analysis (Jolliffe, 1986, PCA), feeding the top 50 components as inputs to the SVM (`SVM/w2v`). The third baseline used FastText embeddings (Bojanowski et al. 2017) in the same way as the word2vec embeddings (`SVM/FT`). The fourth baseline used the vector of counts of CPC subclass appearances in direct citations of a patent as the SVM features (`SVM/1Hop`). The fifth baseline used a combination of the *tf-idf* features from the first model and the citation features from the fourth model (`SVM/tfidf-1Hop`). We tested 8 variants of our architecture, listed at the bottom of Table 3. That table shows which embedding was used (w2v or B4P), and which streams were used with what parameters (all models used stream 1 and 2). For example, `w2v-FullClaims` uses word2vec embeddings throughout, with the full claims for stream 2. On the other hand, `1Hop` used stream 2 with 512 tokens, and incorporated stream 4 with 1-hop citations, and B4P embeddings throughout.

Table 3 List of model variants, showing which embedding is used and which streams they incorporate with which settings

Stream		2	3	4	5
	Word	Claims	Desc.	Cite	CPC
Model	Emb.	Len.	Len.	# Hops	Emb.
`A&F`	w2v	-	-	1-Hot	1-Hot
`A&F/USPTO`	w2v	Full	-	1-Hot	-
`w2v-FullClaims`	w2v	Full	-	-	-
`512ClaimsOnly`	B4P	512	-	-	-
`Plus512Desc`	B4P	512	512	-	-
`1Hop`	B4P	512	-	1	-
`2Hop`	B4P	512	-	2	-
`CPCSeq`	B4P	512	-	-	Seq.
`CPCAvg`	B4P	512	-	-	Avg.
`B4P+All`	B4P	512	512	1	Avg.

All models used stream 1 (abstracts). w2v = word2vec, B4P = Bert for Patents

5 Experiments

5.1 Setup

We trained our models for 5 epochs and in batches of 64 samples, with LSTM hidden unit size 64, dense layers having size 300, 64 and 1, both using 40% dropout. We used the default ADAM optimizer with a learning rate of 0.0001. We performed 5-fold cross validation in all experiments. Experiments using the Full Balanced dataset are shown in Table 4. We also computed learning curves for all models using smaller balanced subsets of the Full Balanced dataset (400, 200, 100, 48, and 24 training examples). Details of these experiments are presented in Section 5.2 and results for the best performing models in Fig. 4. Numbers of parameters for the models ranged from 637,753 (`512ClaimsOnly`) to 1,386,389 (`B4P+All`), which does not include parameters of the pre-trained models. Training each model took approximately two hours on a dual CPU compute node (Xeon Gold 6258R, 2.7 GHz) with 1.5TB RAM and 8 Nvidia A100 40GB HBM2 PCIe 4.0 GPUs. Data processing took approximately 4 hours for each model.

We evaluated several hyperparameter configurations, including variations in the number of hidden units in the LSTM and dense layers, as well as changes to the optimizer and learning rate. Across these trials, the configuration reported by Giczy et al. (2021) consistently achieved the best performance on our validation sets. Therefore, all results presented in this paper were obtained using that configuration detailed above.

Table 4 Average 5-fold cross-validated F_1 scores of all models

		Hard			Easy			Holdout		
Models	Overall	Avg.	+	-	Avg.	+	-	Hard-	Easy+	Easy-
`A&F`	0.68	0.60	0.62	0.57	0.77	0.80	0.74	0.39	0.95	0.43
`A&F/USPTO`	0.73	0.62	0.63	0.60	0.82	0.83	0.82	0.38	0.95	0.79
`Choi`	0.77	0.63	0.63	0.63	0.92	0.91	0.92	0.42	0.95	0.76
`SVM/1Hop`	0.77	0.65	0.60	0.69	0.89	0.89	0.89	0.89	0.94	0.93
`SVM/tfidf-1Hop`	0.77	0.66	0.61	0.69	0.89	0.89	0.89	0.88	0.95	0.93
`SVM/tfidf`	0.78	0.67	0.64	0.69	0.89	0.89	0.88	0.84	0.94	0.9
`SVM/w2v`	0.73	0.67	0.64	0.69	0.79	0.78	0.81	0.55	0.69	0.71
`SVM/FT`	0.78	0.68	0.64	0.68	0.88	0.85	0.91	0.84	0.93	0.90
`w2v-FullClaims`	0.68	0.59	0.54	0.64	0.78	0.75	0.81	0.86	0.81	0.90
`2Hop`	0.76	0.62	0.57	0.67	0.90	0.90	0.90	0.82	0.96	0.88
`CPCSeq`	0.76	0.64	0.65	0.63	0.88	0.89	0.86	0.79	0.98	0.88
`512ClaimsOnly`	0.77	0.66	0.64	0.68	0.88	0.89	0.86	0.81	0.98	0.89
`Plus512Desc`	0.77	0.65	0.63	0.66	0.90	0.90	0.89	0.84	0.98	0.9
`1Hop`	0.77	0.63	0.60	0.66	0.90	0.90	0.90	0.85	0.98	0.91
`CPCAvg`	**0.80**	**0.69**	**0.67**	**0.71**	0.91	0.92	0.91	0.82	0.99	0.92
`B4P+All`	0.79	0.64	0.59	0.70	**0.93**	**0.93**	**0.93**	**0.89**	**0.99**	**0.95**

The **Overall** is the average between **Hard** and **Easy**, excluding the **Holdout** results. The "+" and "-" signs refer to positive and negative classes. Best scores for each column are boldfaced. Standard deviations were 0.32 for `A&F`, 0.29 for `w2v-FullClaims`, and no more than 0.1 for all other models

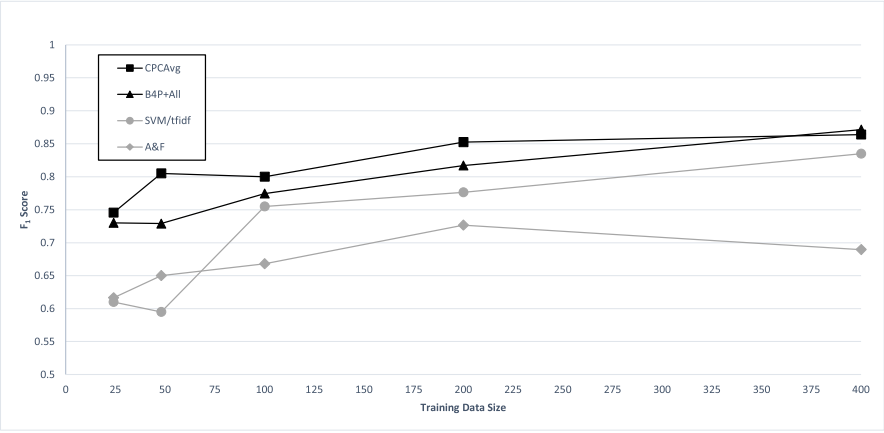

Fig. 4 Learning curve for 400, 200, 100, 48, and 24 training examples on the two best performing models (`CPCAvg` and `B4P-All`) compared with Abood and Feltenberger (2018) model (`A&F`) and the best performing baseline (`SVM/tfidf`)

5.2 Results

As shown in Table 4, `A&F`, `A&F/USPTO`, and `Choi` did not perform nearly as well on our data as originally reported (for example, `A&F` reported an F_1 of 0.98; our experiments resulted in 0.68). Those models also underperformed our baselines. `A&F` and `A&F/USPTO` had low performance both on Hard and Easy examples. One possible explanation for this is our use of balanced data. On the other hand, the SVM models performed quite well both on Easy and Hard examples when *tf-idf* and Word2Vec (or FastText) embeddings were used as feature vectors for abstract and claims and the models were trained on all the data. The SVM models using citation information performed less well than the other baselines, but still reasonably. Using citation information improved performance on the Hard examples. Note that `w2v-FullClaims` is the same as `A&F` (without citation and CPC 1-hot), and shows slightly better performance, which suggests that the 1-hot encodings add noise to the network. Table 4 also shows the performance of the models on the holdout datasets (Table 2). All models except `A&F`, `A&F/USPTO`, and `Choi` perform well on the Holdout data. The best two models on the Full Balanced Dataset are `CPCAvg` and `B4P+All`, the first performing best on Hard examples and the second performing best on Easy examples. `B4P+All` performs the best on the Holdout data.

Figure 4 shows learning curves for our two best performing models (`CPCAvg` and `B4P-All`), the baseline with the best learning curve (`SVM/tfidf`), and Abood and Feltenberger (2018) model (`A&F`), which was the best performing prior model. We trained all models on Balanced Datasets of sizes 400, 200, 100, 48, and 24. For each reduced balanced dataset, we randomly sampled an equal number of examples from each of the four categories, with total per-category sample sizes of 100, 50, 25, 12, 6 examples, respectively. The remaining data were then used to build a single balanced holdout test set for that setting, with the samples per category capped by the number of leftover hard positives. For example, in the 100-per-category setting, the

Table 5 Overall F_1 scores of all models on balanced holdout test sets, 5-fold cross validated on 400, 200, 100, 48, and 24 training examples. Best scores for each column are boldfaced.

Models	400	200	100	48	24
A&F	0.69	0.73	0.67	0.65	0.62
A&F/USPTO	0.75	0.73	0.68	0.66	0.64
Choi	0.75	0.68	0.48	0.48	0.44
SVM/1Hop	0.83	0.80	0.73	0.65	0.61
SVM/tfidf-1Hop	0.83	0.80	0.74	0.62	0.64
SVM/tfidf	0.84	0.78	0.76	0.60	0.61
SVM/w2v	0.69	0.60	0.59	0.57	0.53
SVM/FT	0.67	0.61	0.58	0.54	0.53
B4P-FullClaims	0.85	0.82	**0.82**	0.80	0.72
2Hop	0.84	0.81	**0.82**	0.75	0.68
CPCSeq	0.85	**0.86**	0.78	0.78	0.73
512ClaimsOnly	0.86	0.82	0.78	0.74	0.73
Plus512Desc	0.84	0.83	0.72	0.76	0.71
1Hop	0.84	0.85	0.79	0.77	0.72
CPCAvg	0.86	0.85	0.80	**0.81**	**0.75**
B4P+All	**0.87**	0.82	0.77	0.73	0.73

holdout set contained 295 examples per category since we have a total of 395 hard positives. This holdout set remained fixed across all five folds, while the training data were split for cross validation. The F_1 scores in Table 5 and Fig. 4 are mean scores across the five trained models evaluated on that same holdout set. When data size falls below 100, we see a marked divergence between the baselines and our neural models, reaching a difference of nearly 14 points of F_1 between CPCAvg and B4P-All. Importantly, the neural models show less degradation as the dataset shrinks, with CPCAvg and B4P-All dropping only 14% and 16%, respectively, while the SVM/tfidf drops 27% overall (A&F does not drop very much, but it is not good to begin with). Because we used a distinct balanced holdout set in each experiment, the results illustrate the general pattern of performance change with decreasing training size, rather than strict pairwise comparisons on an identical test set.

6 Discussion

There are several key takeaways from the experimental results. **First**, the prior models A&F, A&F/USPTO, and Choi didn't perform nearly as well on balanced data that includes examples near the landscape boundary. In particular, they perform relatively poorly on Hard examples, showing that examples near the landscape boundary are critical. **Second**, the BERT for Patents (B4P) embeddings were effective in improving performance of architectures in which it was substituted for other embeddings. **Third**, for large amounts of data, most of the models, including the SVM baselines (but excluding A&F), performed quite well and were very close in performance. This suggests that, when a lot of data is available the extra complexity and computational load of the neural methods does not purchase much in the way of performance. **Fourth**, direct citations (1-hop citations) combined with CPC codes give good result on both Easy and Hard examples, but compared with the other features does not seem to add much. Interestingly, 2-hop citations have marginally poorer performance

on the Hard examples. **Finally**, where the neural models really improve over the baselines is in the low-data regime. In the regime of 24 examples (evenly balanced between Hard/Easy and Positive/Negative), we see improvements of nearly 14 points of F_1 over baseline. Interestingly, the `CPCAvg` model, which uses only abstract text, 512 tokens of the claims, and CPC information, is the best overall model. Regardless, even this best model only achieves 0.75 F_1 at 24 examples, which has much room for improvement. It is important to note that, the technology domain we chose for this study, Artificial Intelligence is inherently broad which can simplify the classification task as it covers a wide range of patents with certain high level characteristics. We selected Artificial Intelligence as the study domain for several reasons. First, our collaborators in the USPTO were tasked with developing a landscaping model for AI, which was the proximal reason for the selection. Secondarily, we had easy access to expertise in AI through our research group. Third, the AI domain was similar in topic and broadness to the "machine learning" domain explored in the original Abood & Feltenberger paper, which made it a good comparison. Note that despite the fact that the domain is broad, and potentially is "easier" than a more specific domain, our experimental results show that the task is still challenging for the methods explored.

7 Limitations & future work

There are several limitations of this work. First, we only examine a single patent landscaping domain, that of AI. AI is a fairly broad technology area and it is not clear that the results will generalize to more specific landscape topics. Second, there may of course be other neural architecture styles that work better; we didn't exhaustively explore these choices as we felt they were beyond scope of the paper as we envisioned it. Future work should explore the space of possible architecture designs. Third, citation network information did not improve the results as much as expected, which is counter-intuitive. We suspect that there are ways to use this information that will improve performance even with the other streams of information, but we were unable to identify them. In addition, in our small-data experiments (e.g. 100, 48, or 24 examples), we constructed balanced datasets through random sampling and then applied 5-fold cross validation. While this allows us to estimate performance variance across folds, the results may still be sensitive to the particular sampled examples included in these small datasets since we did not construct multiple independent datasets of the same size. Future work should address this limitation. Another limitation concerns the use of our annotation tool, *Patentify*, for active learning. While active learning is a fairly established technique, and *Patentify* has not been presented elsewhere, in this study we validated its effectiveness indirectly through the use of the annotated data obtained using the tool. Furthermore, while we use the term "hard" to refer to example near the bag-of-word SVM classifier, we did not conduct a separate evaluation of these examples to confirm that they were truly "hard", either for the neural approaches or for people. Thus, our approach assumes that active learning was successful in prioritizing informative examples near the boundary. Finally, the performance of the model in the small data regime could still be improved; it remains

to be seen what the true lower limit of data is required to construct a good landscape for such a broad technology domain.

There are a number of potential directions for future work. While BERT for Patents is the only patent-adapted embedding model available, it would be interesting to try more recent embeddings models focused on larger spans of text (sentences, paragraphs, sections), with or without adapting to the patents domain. Second, we do not explore any generative approaches at all, which seem to be quite powerful and flexible. Future patent classification systems should most certainly investigate these possibilities.

8 Contributions

In this paper, we have demonstrated the performance of various neural models on a patent dataset that we have built. Our work contributes to the research of automated patent landscaping models in four different aspects. First, we have shown that previously proposed "seed/anti-seed", while useful, importantly lacks examples near the decision boundary and using only seed/anti-seed data gives a misleading view of model performance. Second, we demonstrate an active learning augmentation to the seed/anti-seed approach which can quickly generate high-quality data near the decision boundary. Third, we conducted systematic experiments comparing the utility of different portions of the information in a patent, concluding that abstract, claims, and CPC codes (and not description or citations) provide the most power. Fourth, when a lot of data are available (1000s of examples) simple methods like SVM work just as well as the neural architectures with different features not making much difference. Fifth, we have experimented with using citation information (1-hop and 2-hop) and showed that this information does not add much performance beyond using text in the claims. Finally, we show that the neural methods significantly outperform the baselines in the small data regime (less than 100 total training examples).

Acknowledgements This work was funded by the National Science Foundation via IIS-1749917. We gratefully acknowledge valuable discussions with Nicholas Pairolero, Alexander Giczy and Gerard Torres from the USPTO. We also acknowledge the contributions of all the undergraduate students at Florida International University who were involved in building the annotation tool, Patentify: Rahul Mittal, Walter Bozzetti, Daniel Perez, Angelo Molina-Rossi, Mia Gabb, Angel Saldivia, William Smith, Ronald Pena, Manuel Garcia, Simarjeet Singh, Carlos Alfredo, Eric Tu, Jordy Araujo, Luis Gil, Katalina Keyser, Josecarlos Lusbel, Samuel Muvdi, Freddy Somarriba, Miguel Santana, Leonel Santos, Jack Brett, Sebastian Galloway, Michael Matos, John Gonzalez, Juan Sanchez. We also acknowledge our annotators: Akul Singh, Ximena Puig, Rosemarie Crespo, Anurag Acharya, Antonela Radas, Amanda Chacin-Livinalli, Ibrahim Alyami, Ana Carolina Oliveira, Mustafa Ocal, Anshu Kiran Sharma, Diego Castro Estrada, Kayla Christopher, Azwad Anjum Islam, Tisa Islam Erana.

Author Contributions Tisa Islam Erana: Methodology, Software, Investigation, Resources, Data Curation, Writing – Original Draft. Mark Finlayson: Conceptualization, Writing – Review & Editing, Supervision, Project Administration, Funding Acquisition.

Funding This work was funded by the National Science Foundation via IIS-1749917.

Data Availability To support reproducibility and extension of the work, we release our code and data in standard formats via our institutional repository at https://doi.org/10.34703/gzx1-9v95/QDLKVW.

Declarations

Conflicts of Interest The authors have no financial or nonfinancial interests that are directly or indirectly related to the work submitted for publication.

Ethical Approval Not applicable.

References

Abood A, Feltenberger D (2018) Automated patent landscaping. Artif Intell Law 26:103–125

Antonin B, Cyril V (2023) Identifying technology clusters based on automated patent landscaping. PLoS ONE 18(12):1–17 https://doi.org/10.1371/journal.pone.0295587.

Beltagy I, Lo K, Cohan A (2019) SciBERT: A pretrained language model for scientific text. In: Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP). Association for Computational Linguistics, Hong Kong, China, pp 3615–3620. https://doi.org/10.18653/v1/D19-1371. https://aclanthology.org/D19-1371.

Bojanowski P, Grave E, Joulin A et al (2017) Enriching word vectors with subword information. Trans Ass Comput Linguist 5:135–146.https://doi.org/10.1162/tacl_a_00051.

Boser BE, Guyon IM, Vapnik VN (1992) A training algorithm for optimal margin classifiers. In: Proceedings of the 5th Annual Workshop on Computational Learning Theory (COLT '92). Association for Computing Machinery, New York, NY, USA, COLT '92, p 144–152.https://doi.org/10.1145/130385.130401.

Choi SJ, Lee H, Park E, et al (2019) Deep patent landscaping model using transformer and graph embedding. arXiv: Computation and Language

Devlin J, Chang M, Lee K, et al (2018) BERT: pre-training of deep bidirectional transformers for language understanding. CoRR. arXiv:1810.04805

Fleiss J (1981) Statistical methods for rates and proportions, 2nd edn. John Wiley & Sons, New York, NY

Giczy AV, Pairolero NA, Toole AA (2021) Identifying artificial intelligence (ai) invention: a novel ai patent dataset. J Technol Transf 47:476–505

Hochreiter S, Schmidhuber J (1997) Long Short-Term Memory. Neural Comput 9(8):1735–1780 https://doi.org/10.1162/neco.1997.9.8.1735. https://direct.mit.edu/neco/article-pdf/9/8/1735/813796/neco.1997.9.8.1735.pdf.

Hunt D, Nguyen L, Rodgers M (2007) Patent Searching: Tools & Techniques. Wiley, https://books.google.com/books?id=iXpNx4n1fDwC.

Jolliffe I (1986) Principal Component Analysis. Springer Verlag

KISTA (2023) KISTA patent trends reports. https://biz.kista.re.kr/patentmap/front/common.do?method=main. Accessed 14 Jul 2023

Landis JR, Koch GG (1977) The measurement of observer agreement for categorical data. Biometrics 33(1):159–174

Lee JS, Hsiang J (2020) Patent classification by fine-tuning bert language model. World Patent Information

Lewis DD, Gale WA (1994) A sequential algorithm for training text classifiers. In: Croft BW, van Rijsbergen CJ (eds) SIGIR '94. Springer, London, London, pp 3–12

Li S, Hu J, Cui Y et al (2018) Deeppatent: patent classification with convolutional neural networks and word embedding. Scientometrics 117:721–744

Li X, Chen H, Zhang Z, et al (2007) Automatic patent classification using citation network information: an experimental study in nanotechnology. In: JCDL '07

Mikolov T, Chen K, Corrado G, et al (2013) Efficient estimation of word representations in vector space. CoRR arXiv:1301.3781. http://dblp.uni-trier.de/db/journals/corr/corr1301.html#abs-1301-3781.

Pujari S, Strötgen J, Giereth M, et al (2022) Three real-world datasets and neural computational models for classification tasks in patent landscaping. In: Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing. Association for Computational Linguistics, Abu Dhabi, United Arab Emirates, pp 11498–11513. https://aclanthology.org/2022.emnlp-main.791.

Pujari SC, Friedrich A, Strotgen J (2021) A multi-task approach to neural multi-label hierarchical patent classification using transformers. In: European Conference on Information Retrieval

Rob Srebrovic JY (2020) Leveraging the bert algorithm for patents with tensorflow and bigquery. https://services.google.com/fh/files/blogs/bert_for_patents_white_paper.pdf.

Seneviratne D, Geva S, Zuccon G, et al (2015) A signature approach to patent classification. In: Zuccon G, Geva S, Joho H, et al (eds) AIRS, Lecture Notes in Computer Science, vol 9460. Springer, pp 413–419. http://dblp.uni-trier.de/db/conf/airs/airs2015.html#SeneviratneGZFC15.

Settles B (2012) Active Learning. Synthesis Lectures on Artificial Intelligence and Machine Learning, Morgan & Claypool Publishers

USPTO (2019) PatentsView. https://www.patentsview.org/download/. updated December 31, 2019

USPTO (2022) FY 2022 workload tables. https://www.uspto.gov/about-us/performance-and-planning/uspto-annual-reports. Accessed 10 Jul 2023

Yun J, Geum Y (2020) Automated classification of patents: A topic modeling approach. Comput Indust Eng 147:106636

Zhang W, Yoshida T, Tang X (2011) A comparative study of tf*idf, lsi and multi-words for text classification. Expert Syst Appl 38(3).https://doi.org/10.1016/j.eswa.2010.08.066.

Authors and Affiliations

Tisa Islam Erana[1] · Mark A. Finlayson[2]

✉ Tisa Islam Erana
tisla016@fiu.edu

Mark A. Finlayson
markaf@fiu.edu

[1] Knight Foundation School of Computing and Information Sciences, Florida International University, 11200 S.W. 8th Street, Miami, Florida 33199, USA

[2] Knight Foundation School of Computing and Information Sciences, Florida International University, CASE Building, Room 362, Miami, Florida 33199, USA