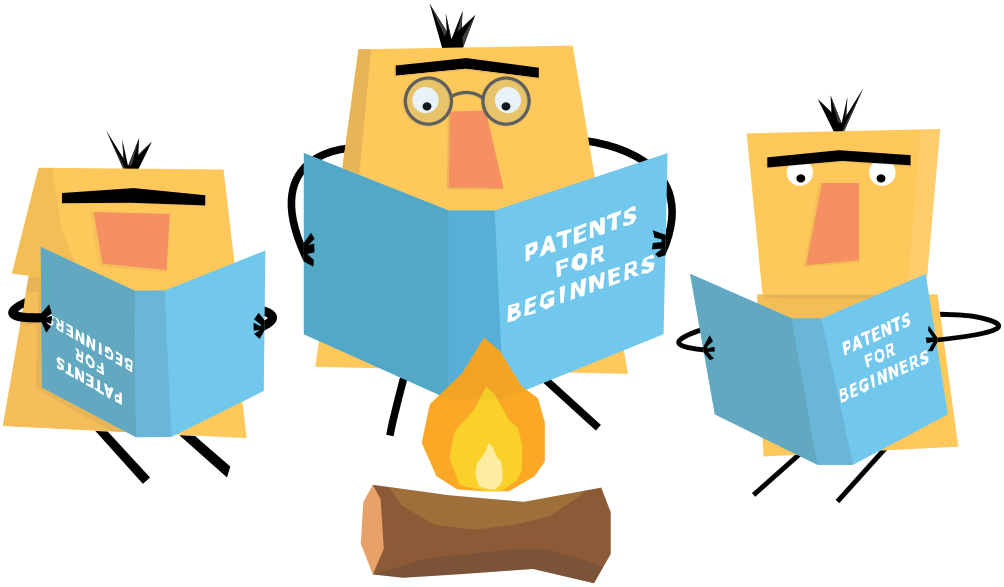
CHALMERS
UNIVERSITY OF TECHNOLOGY



Exploring Supervision Levels for Patent Classification

Using Weakly Supervised, Semi-supervised, and Supervised Learning to Classify Patents at Different Granularity Levels

Master's thesis in Data Science and AI

ADAM VAN HOEWIJK AND HENRIK HOLMSTRÖM

DEPARTMENT OF MATHEMATICAL SCIENCES

CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2022
www.chalmers.se

Master's thesis 2022

Exploring Supervision Levels for Patent Classification

Using Weakly Supervised, Semi-supervised, and Supervised Learning to Classify Patents at Different Granularity Levels

ADAM VAN HOEWIJK and HENRIK HOLMSTRÖM

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Mathematical Sciences
Division of Data Science and AI
Chalmers University of Technology
Gothenburg, Sweden 2022

Exploring Supervision Levels for Patent Classification
Using Weakly Supervised, Semi-supervised, and Supervised Learning to Classify Patents at Different Granularity Levels
ADAM VAN HOEWIJK and HENRIK HOLMSTRÖM

Supervisor at CSE: Dana Dannélls, Computer Science and Engineering
Examiner at CSE: Marina Axelson-Fisk, Mathematical Sciences

Master's Thesis 2022
Department of Mathematical Sciences
Division of Data Science and AI
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Cover: BERT & Co. learning how to classify patents.
Figures containing dogs or cats use resources from Flaticon.com

Typeset in LaTeX
Printed by Chalmers Reproservice
Gothenburg, Sweden 2022

Exploring Supervision Levels for Patent Classification
Using Weakly Supervised, Semi-supervised, and Supervised Learning to Classify Patents at Different Granularity Levels
ADAM VAN HOEWIJK AND HENRIK HOLMSTRÖM
Department of Mathematical Sciences
Chalmers University of Technology

Abstract

Machine learning can help automate monotonous work. However, most approaches use supervised learning, requiring a labeled dataset. The consulting firm Konsert Strategy & IP AB (Konsert) sees great value in automating its task of manually classifying patents into a custom technology tree. But the ever-changing categories leaves a pre-labeled dataset unavailable. Can other forms of supervision be used for machine learning to excel without extensive data? This thesis explores how weakly supervised, semi-supervised, and supervised learning can help Konsert to classify patents with minimal hand-labeling. Furthermore, what effect class granularity has on performance is explored alongside whether or not using patents' unique characteristics can help.

Two existing state-of-the-art methods at two supervision levels are employed. Firstly, LOTClass, a keyword-based weakly supervised approach. Secondly, MixText, a semi-supervised approach. We also propose LabelLR, a supervised approach based on patents' cooperative patent classification (CPC) labels. Each method is tested on all granularity levels of a technology tree provided by Konsert alongside a combined ensemble of the three methods. MixText receives all unlabeled patent abstracts together with the same ten labeled documents per class LabelLR receives. LOTClass on the other hand receives the unlabeled abstracts along with class keywords.

Results reveal that the small training dataset of around 4 200 patents leaves LOTClass struggling while MixText excels. LabelLR outperforms MixText on the rare occasion when the CPC labels and the classifications closely match. The ensemble proves more consistent than LabelLR but only outperforms MixText on some granular classes. In conclusion, a semi-supervised approach appears to be the best balance of minimal manual work and classification proficiency reaching an accuracy of 60.7% on 33 classes using only ten labeled patents per class.

Keywords: Patent, Weakly supervised learning, Semi-supervised learning, Supervised learning, BERT.

Acknowledgements

We would like to thank Konsert for providing us with the resources and opportunity to work on this thesis along with great workout sessions and breakfasts every Tuesday. A special thank you to Emil Haldorson and Jonas Lindgren from Konsert for their support. We also want to thank Lisa and Mats Holmström, and Mia van Hoewijk for proofreading. A big thank you to our supervisor Dana Dannélls for continuously offering input on the work as well as answers to any questions or concerns. Additionally, we thank Marina Axelson-Fisk for being our examiner for the thesis.

Adam van Hoewijk and Henrik Holmström, Gothenburg, June 2022

List of Acronyms

This is a list of alphabetical ordered acronyms used throughout this thesis:

BERT	Bidirectional Encoder Representations from Transformers
CPC	Cooperative Patent Classification
IP	Intellectual Property
IPC	International Patent Classification
MLM	Masked Language Model
MCP	Masked Category Prediction

Contents

List of Figures

List of Tables

1
Introduction

What is the mark of great language-focused artificial intelligence? Spectacular displays of language understanding? A nuanced vocabulary? We argue applicability. A text classifier trained on plentiful labeled example texts can display great language understanding. However remarkable, such a model suffers from limited utility. In practice, labeling example data is time-consuming and labor-intensive. A machine learning model that requires little supervision, that is, little or no manually labeled data offers a solution. An approach that finds insight in unlabeled examples opens the door to a wast ocean of practical applications. One such application is the categorizing of patents.

Konsert is a niche consulting firm with a focus on technology innovation and intellectual property (IP). As part of their work developing business-driven IP strategies for industry clients, Konsert has the recurring task of analyzing external IP landscapes. Their most common approach involves sorting patents into manually pre-defined frameworks. Firstly, this allows for the mapping of opportunities and constraints of a technology field. For example, which parts of the field are less protected and where investments are being made. Secondly, it gives Konsert a framework of patents they can refer back to when consulting with a client. However, reading and categorizing patents manually is time-consuming and limits the amount of data Konsert can process. Every time Konsert analyzes a new IP landscape a new framework is created and the manual patent sorting starts once more. Automating the monotonous work would enable Konsert to use more comprehensive patent datasets consequently increasing the scope and quality of client insights.

For a machine learning model to accurately classify text requires a nuanced, deep understanding of natural language. State-of-the-art methods show an understanding of context and complex language, classifying everything from news articles to movie reviews [36]. But for Konsert, patents offer a unique challenge. The vocabulary in patents is domain-specific [37] and the language is ambiguous [7]. Such language leaves most language models struggling. Extensive data-consuming model training is thus vital. Pre-trained models offer a unique opportunity in having profound language understanding with training done on general tasks [15]. A pre-trained model can give way to proficient classification with little to no labeled data [51, 47, 28, 36]. But the question of whether or not these models are capable of navigating the complex landscape of patent classification remains open.

Deviating from supervised learning where a labeled dataset is required, semi-supervised

and weakly supervised learning offer an opportunity. Two approaches that can prove successful in classification even when deprived of labeled data. Several methods have been developed [51, 47, 28], but they are yet to be applied to patent classification. Expanding upon some of these methods to adapt to patents' unique characteristics can reveal what level of supervision is needed to help a company such as Konsert.

1.1 Aim

This thesis aims to classify patents with different levels of supervision utilizing patents' unique characteristics to elucidate when adequate proficiency is reached. These aims can be divided into the following questions:

- What supervision level fits best for patent classification while minimizing manual work?
- What performance can be achieved in patent classification?
- How does classes' granularity affect performance in patent classification?
- How can patents' unique properties be utilized in patent classification?

1.2 Outline

Now that the problem has been introduced and the aims have been stated the rest of the thesis follows the subsequent outline. Chapter 2 gives a background to the necessary theory for a complete understanding of the thesis. Including supervision levels, machine learning models, and how patents work and are written. Following Chapter 2, Chapter 3 gives background to the datasets to be used in the method, described in Chapter 4. In Chapter 5 we show the results which are then discussed in Chapter 6. Finally, a conclusion is drawn in Chapter 7.

1.3 Rationale

For each aim to be reached a rationale for achieving the aims of the thesis is presented. To test what supervision level fits best several approaches are trained on the dataset. The different models are compared both regarding supervision level and predictive performance to evaluate what fits best. A model is also fine-tuned on the full dataset to see what proficiency is possible. To understand how the classes' granularity affects performance the dataset is tested on the different levels of its hierarchical structure. To see how unique aspects of patents can be used a method will be tested that classifies using patents' CPC labels.

1.4 Scope

Because of the constrained time frame of the thesis, the scope is limited. Therefore, when it comes to the aims of the thesis they can only be investigated to a limited extent. Only three different methods are tested. Text classification is limited to the abstract and description and does not include parts such as claims. The other

attribute of patents that is looked at is limited to the predefined hand-labeled CPC labels included in the patents. Additionally, only one dataset from Konsert is used for analysis.

1.5 Limitations

The thesis is technically limited. Computing power is limited in Google's Colaboratory service which means that entire patents cannot be used. Instead selected parts of the patents are chosen. Limited computing power also limits choices such as the number of epochs reasonable to run.

1.6 Ethical Considerations

One of the first ethical considerations is that a machine learning model can adopt biases. A bias can cause a model to tend to categorize based on aspects that are not relevant to the category. This leads to certain patents being systematically put into the wrong category. If this machine learning is used in patent classification one could imagine, for example, Chinese patents tending to be left incorrectly classified. Possibly due to a different grammatical structure. Considering that the patent classifications are intended to be used to analyze different technology areas, failing to classify a certain country's patents correctly could lead to unfavorable consequences. A bias could come from the data used, the processing of that data, the model's structure, and the keywords or example patents used for each category [48].

A second ethical consideration is a model's impact on people's work. Instead of supplementing people's jobs and making them easier it could replace their job. Although making society more productive is generally desirable, in the short term, someone could be left without a job.

1.7 Contributions

First and foremost we contribute by testing state-of-the-art weakly supervised and semi-supervised methods on patents. Furthermore, we test how well the CPC labels can be used to classify other categories than the CPC labels' own category. Finally, we evaluate how a combination of several approaches can affect predictive performance along with what effect class granularity has on each approach.

2
Background

This chapter explains some essential theory regarding patents and approaches to machine learning.

2.1 Patents

Some of companies' most valuable assets can be IP. IP is concepts or ideas one has created that can be legally protected. For example, a manufacturing technique. To protect one's IP a patent can be issued. A patent is a legal right to that IP. The benefits of a patent are fairly straightforward. If you invent something you can legally protect it and stop others from copying by suing. It does come with limitations though. A patent only gives protection for a limited amount of time, generally 20 years. This means that you can get a head start without any competition given no one infringes on your patent. Owning a patent can be valuable, but having a patent issued relies on filling some criteria. It must be something new. Meaning you are the first to do it. It must be non-obvious and it must have utility. If it can not be of use there is no point to the patent [7].

2.1.1 Contents of Patents

In practice, patents follow a standardized structure. This makes sure one can interpret what the patent protects and what the invention is. To start with, there is a title and an abstract. The abstract gives the reader a summary of what the patent contains [7]. Technical details can be included [19] but it primarily serves to give a more general idea of the patent's contents. In case one wants a more detailed understanding further exploration of the patent will be in the description.

After the abstract, one finds the description. The description is considerably longer than the abstract but also includes more technical details. The exact contents of the description can vary but in general, the invention must be described in enough detail that the invention can be used by anyone skilled in the technical field. Parts of the description can include a background to the problem, examples of where and how the invention can be used, and technical features. Furthermore, figures can be used. This is done in combination with the text to further illustrate ideas [7]. The description provides insights but does not determine the legal protection of the patent. This is instead done in the claims sections.

Possibly the most important part of a patent is the claims. The claims is a list of specifications determining what the patent legally protects. Other parts of the patent can provide more insight but the claims are what will protect you in case of an infringement on your patent. In case you sue someone whom you believe is using your IP, a judge will examine the claims to determine whether or not you are protected. Claims along with the description and abstract make up most of the text within a patent. But several other pieces of information are also included. Examples include the inventor, the owner, related patents, and interestingly classifications.

Another important part of a patent is its classifications. Patents are manually classified into different technology areas. This helps one find patents relevant to a given technology. There have been several classification schemes one example being the International Patent Classification (IPC) [22]. Although IPC has been commonly used, in 2013 the European Patent Office and the US Patent and Trademark Office jointly finished creating the Cooperative Patent Classification, in short CPC. [1]. The CPC is similar to the IPC. It is a classification scheme where the categories are hierarchically ordered into different technologies. For example the class *H01L27/0744*. The *H* stands for electricity. The *H01* is electric elements. *H01L* is semiconductor devices. In short, one can search for patent classes with different granularity. A patent can be part of a very specific class but the first letters and numbers represent a broader category [7]. The example category's different parts are illustrated in Figure 2.1 with the broadest category being its section, followed by class, sub-class, group, and sub-group.

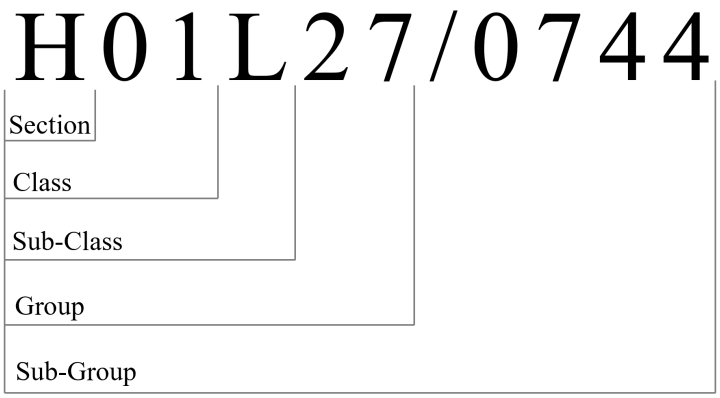


Figure 2.1: The different parts of a CPC label.

2.1.2 Patent Language

For obvious reasons a patent that gives broad protection is of more use. Writing specific claims leaves the protection equally specific [7]. Language formulated in a way that allows for several interpretations can mean a patent can be interpreted to protect more. Furthermore making a description too descriptive can make a competitor understand your invention and opens the door to copycats [7]. Because of this, several challenges arise with patents' language. Firstly, the sentence length is significantly longer than in more general corpora. Secondly, the vocabulary is

unconventional with domains specific terms. Finally, the syntactic structure is complex [46]. These factors mean that patents' language is hard to read and understand requiring excellent language understanding and a deep vocabulary.

2.1.3 Patent Landscaping

A patent can provide great value but also comes at a cost. Legals fees for a patent exceed thousands of euros [44]. Since patents are publicly available, patent investments provide insight into what areas of technology are being invested in. Looking at a patent's bibliographic information can provide even more insight into the investments. What type of patents are issued in what countries can tell you where investments are being made as well as what inventors are at the cutting edge their respective field. The practice of analyzing patents in this way is called patent landscaping [43].

Creating patent landscapes can be a great way for a company to look for opportunities and threats to one's business. On the flip side, it can be a labor-intensive task. Using the existing patent classifications can be a shortcut to finding patents in your area of interest but rarely will you find a category matching your exact needs. This is where companies can spend extensive time and resources reading and classifying patents into custom categories for their specific needs [43]. A task not helped by the difficult language patents so often contain.

2.2 Levels of Supervision in Machine Learning

There are several approaches to machine learning with different levels of supervision being one way in which they vary. More labeled data or more help can be considered a higher level of supervision. Unsupervised learning lies on one side of the spectrum while supervised learning is on the other [11]. In between these levels, one can find alternatives that don't provide as much supervision as supervised learning but more than nothing as in the case of unsupervised learning. Some of these approaches are seen in Figure 2.2. Weakly supervised [42] and semi-supervised learning [52] are examples that fit in between the two extremes. Each one of these methods is further explained in the following section.

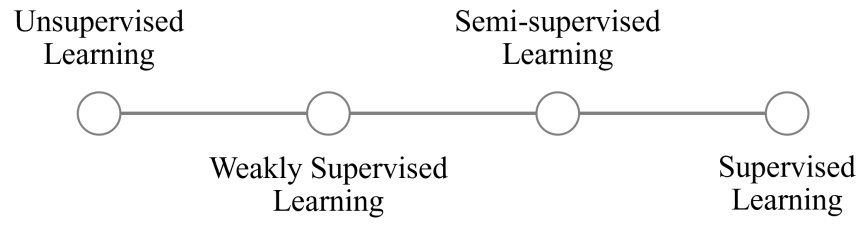


Figure 2.2: Different supervision levels on a spectrum.

2.2.1 Supervised Learning

One of the most robust machine learning approaches is supervised learning [13]. In supervised learning, data is annotated by mapping a piece of data to a target variable. A model is then trained on the annotated examples. Practically this can be done by having a feature vector X and assigning it a variable y [38]. If one wants to train a model in binary classification, the target variable would be a 1 or 0 representing whether or not the data belong to a class [6]. The piece of data can also be assigned a continuous variable. Using machine learning to predict a continuous variable is instead called regression [14]. As an example of classification, imagine having pictures of cats and dogs. After manually labeling the picture with cat or dog a machine learning model can then be trained on these examples to classify cat and dog pictures as illustrated by Figure 2.3.

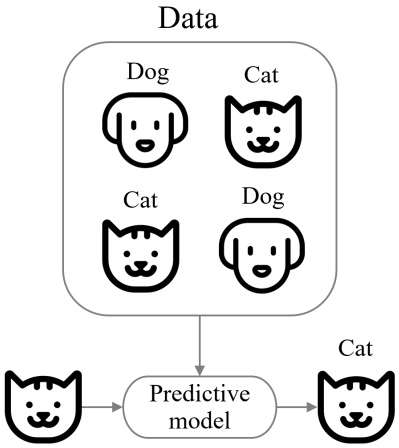


Figure 2.3: Example flowchart of a supervised learning.

Common examples of supervised classification approaches include logistic regression which models the probability that a piece of data belongs to a certain class [27] and random forests which is a combination of multiple decision trees [30]. Another example is neural networks which can classify a broader range of data [49].

2.2.2 Semi-supervised Learning

Semi-supervised learning lies between supervised learning and unsupervised learning. In semi-supervised learning, you have access to some labeled data and a large amount of unlabeled data. There are several ways to use the unlabeled data. For example, the labeled data can be used to generate more labeled data from the unlabeled data consequently combining the two. One way of doing this could be to assign the same label as the closest labeled data point. If your data points are feature vectors you can simply check the distance between the vectors [38]. Another example could be if you have some labeled texts describing cats and dogs and many unlabeled cat and dog texts. By counting how many words the unlabeled texts share with the cat and dog texts one could label some of the unlabeled texts. Simply compare what texts share the most words with either the cat or dog texts. These now

labeled examples can be used to train a more sophisticated model using supervised learning as illustrated in Figure 2.4.

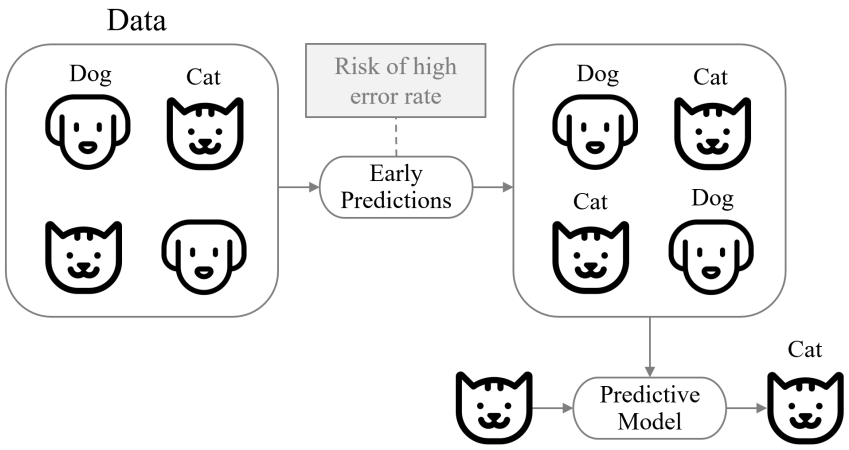


Figure 2.4: Example flowchart of a semi-supervised learning.

2.2.3 Weakly Supervised Learning

Weakly supervised learning can be a path to good prediction performance with little manual work [42]. Instead of using labeled data, it uses other sources of data that are easier to come by. This data can then be used to create a dataset that can be utilized in conventional supervised learning. An example of this would be providing keywords for each class [42]. Imagine having texts about dogs and cats. You can then provide keywords for each class, for example, the class names 'dog' and 'cat'. Each text containing the word 'dog' can then be labeled as a dog text and texts containing the word 'cat' can be labeled as cat texts. The now labeled data can be used to train a model that is more complex than simply searching for the word 'dog' or 'cat'. This allows us to classify texts that do not contain any keywords or maybe contain both keywords equally. The concept is illustrated by Figure 2.5.

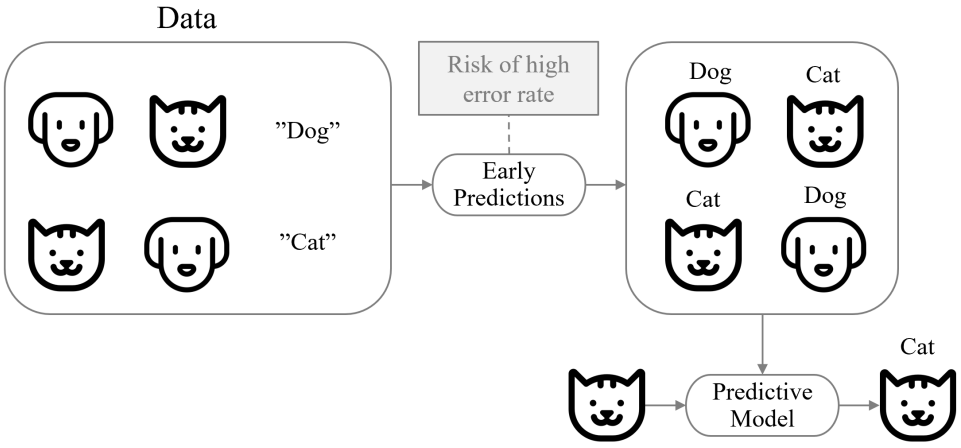


Figure 2.5: Example flowchart of a weakly supervised learning.

2.2.4 Unsupervised Learning

Unsupervised learning requires no manual annotation. Instead, it seeks to find natural patterns in the data. Either by finding clusters and assigning a label or by assigning values and ranking the data [38]. An example of this could be having a bunch of pictures of cats and dogs, and using machine learning to find natural clusters as shown in Figure 2.6. The clusters may or may not represent cats and dogs. Any other pattern in the data could be found. Furthermore, the label of each cluster would simply be a cluster id and not animal names.

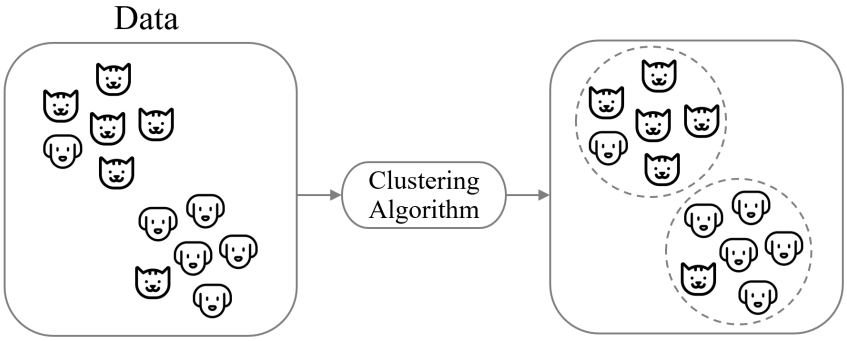


Figure 2.6: Example flowchart of a unsupervised learning.

A common approach is K-means clustering which tries to find the middle of clusters by moving randomly initiated centroids in the direction of clusters. By iteratively assigning data points to the closest centroid and moving the centroid to the center of the assigned data points, k-means clustering can find natural clusters in the data [24]. Another common approach is DBSCAN which instead clusters based on data density [18].

2.3 Machine Learning Approaches

In the following section, a few machine learning methods are explained.

2.3.1 Language Modeling

Statistical language modeling, or more simply language modeling is a method where the goal is to draw insight from sequences of words through the use of statistics [2]. Language modelling is a general term used to describe statistical models that take text as an input. This could for example be a model that ranks words after how important they are perceived to be in a text as illustrated in Figure 2.7.

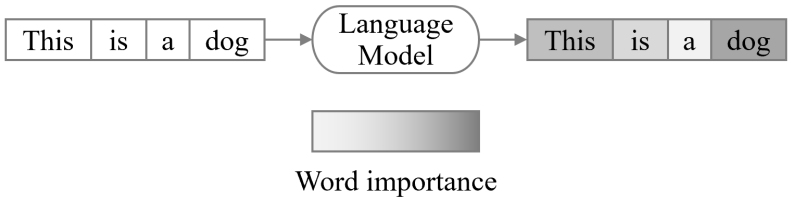


Figure 2.7: Example language model.

The first Language models were simple like Ziph's that explored statistical patterns between how word occurrence decays as a power function of a word's rank [12]. Later examples of Language models are more involved such as the transformer architecture [45].

2.3.2 Transformers

Transformers are the base of several recent natural language processing techniques such as BERT [15]. Transformers consist of two parts, an encoder, and a decoder which are the left and, right blocks respectively as shown in Figure 2.8 [45]. An encoder can not simply read a text such as 'This is an dog', it has to be tokenized first. Tokenizing a text means that you exchange each word with its index in a predefined list of words to create a number representation. The text 'This is a dog' could be converted into the tokens ['This', 'is', 'a', 'dog'] which could have the corresponding numbers in the list of tokens [1, 2, 3, 4]. This tokenized text can be introduced to the encoder where it additionally saves the position of the words in the sentence. From this point forward the goal of the encoder is to take this text and create its attention. Attention refers to the encoder's perceived importance of each word in the text relating to the rest of the words. For the earlier text, this might mean that 'dog' would be given a higher value than 'a' because it is of higher importance to the sentence's meaning. Similar to the case in Figure 2.7. The encoder in this sense simply says what words in a text contextually should be paid more attention to.

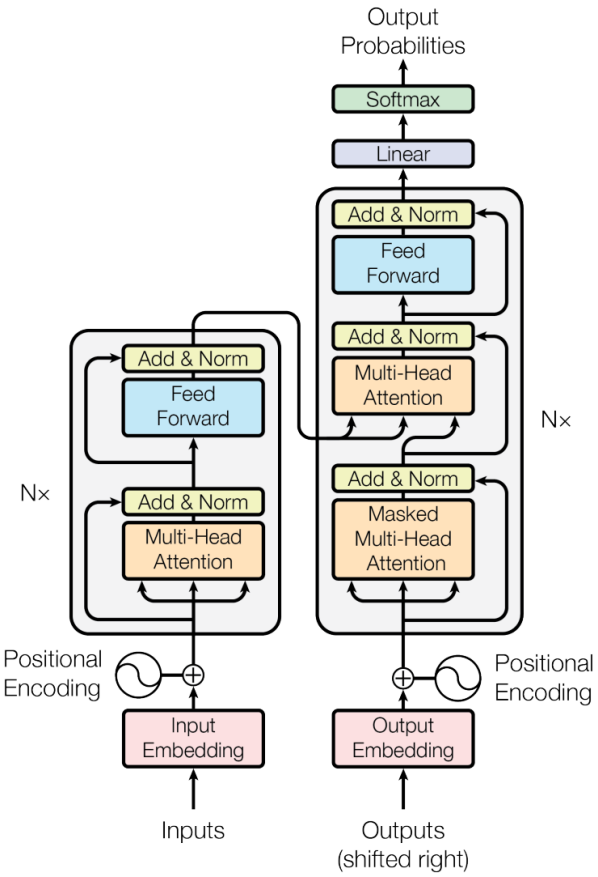


Figure 2.8: Model architecture of a transformer [45].

The decoder can be thought of as writing a text one word at a time, where the next word is determined by the previous words written and the importance of the words from a text given to the encoder. To be precise, the goal of the decoder is to determine the next token of the output by using a few inputs. These inputs are all the previously predicted tokens and the attention of all tokens inserted into the encoder.

2.3.3 BERT

BERT, the acronym of Bidirectional Encoder Representations from Transformers is made from one part of the transformer architecture, the encoder. Only the encoder is needed since the goal of BERT is to create a language model, such as assigning the attention of a sentence. BERT is often called bidirectional though non-directional might be a more intuitive description. This comes from the fact that BERT does not read a sentence from left to right as other models might do, instead, BERT looks at the complete sentence at once when determining its context. This is done with the help of three separate embeddings BERT receives from the tokenizer seen in Figure 2.9. The position embedding describes the position of each token, the segment embedding describes if the text is separated, and the token embedding consists of the input tokens. To increase the complexity of what a BERT model can understand it actually consists of several such attention mechanisms. First of all, there are several

mechanisms that work next to one another called *heads*. Secondly, the output of the attention mechanism are feed into several layers of attention mechanisms called *hidden layers*. By varying the number of attention heads and layers one can form BERT models of different sizes.

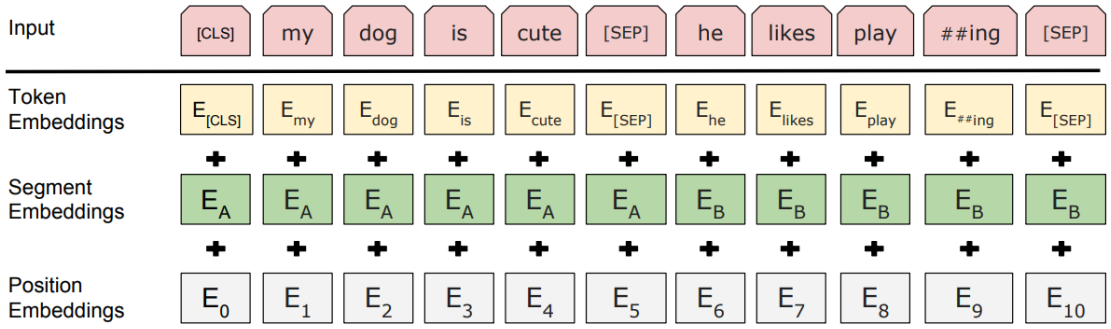


Figure 2.9: The three parts of the BERT input embeddings. The token embeddings, the segmentation embeddings and the position embeddings [15].

There are several pre-trained BERT models. The earliest model, $BERT_{base}$, is trained on a books corpus (800M words) and the English Wikipedia (2,500M words) [15]. This pre-training means that only fine-tuning of a BERT model's parameters is required to adapt it to a specific task. Before examining how BERT can be fine-tuned it is helpful to have an understanding of how the pre-training is done.

When training a language model a prediction goal needs to be determined. BERT uses two training strategies. The first one is called Masked Language Modeling which means that some words in a text are replaced with the [MASK] token and BERT tries to predict the original value of that token based on the surrounding tokens. This can be compared to covering up a word in a text and making BERT predict based on the surrounding words. Masked Language Modelling makes BERT good at using context to understand words.

The second training method is called Next Sentence Prediction. The goal of Next Sentence Prediction is to predict whether two sentences come after each other or are two random sentences. For this task, a [CLS] token is inserted at the beginning and a [SEP] token after each sentence. From the [CLS] token the similarity of the sentences can be extracted.

When fine-tuning BERT for classification the [CLS] token can be used as an output to understand what words are important in a specific classification problem. Different words can have changing levels of importance in different classification tasks. Because of this, the models' parameters have to be fine-tuned for the [CLS] token to aid in classification for specific tasks [15].

2.3.3.1 BERT Variations

There exist different variations of the original $BERT_{base}$ model each made for different purposes [16]. One example is MobileBERT, a BERT model more than five times smaller than the base model. This allows for similar performance to a $BERT_{base}$ model but requires less computing power [40]. Another variant is $BERT_{large}$. This model is instead larger than the original with more parameters to tune. Because of the size, it is heavier to run but can also learn more [16]. Such a model thus fits for harder tasks such as reading patents. A variant of the $BERT_{large}$ model is *BERT for patents*. A model trained specifically on patents [33]. *BERT for patents* is trained on the text of over 100 million patents [39]. This entails a few things. First of all, it means that the model is adapted to the peculiar language in patents. This can help the model understand complex language specific to patents. Secondly, it means that the model's vocabulary is adapted to patents. Instead of prioritizing the words most commonly used in everyday language the *BERT for patents* model can include words in its vocabulary commonly used in patents language. Additionally *BERT for patents* has a few extra tokens indicating what part of a patent text comes from. Examples include the tokens [TITLE] and [ABSTRACT]. This does make the model more niche but also allows it to excel further than other models in the domain [33]. After the release of BERT, models such as RoBERTA which introduces better pre-training have shown significantly better performance [25]. Still, *BERT for patents* has shown even better results when it comes to domain-specific tasks in patent classification [5].

2.3.4 Logistic Regression

Logistic regression is a regression methodology used for binary classification. Given any input features Logistic regression models the probability of an output of one or zero [34]. To achieve this a logistic regression model has a tunable weight for each input feature. The input and the weights are multiplied to be run through a logistic function. The logistic function essentially compresses the output of the model to be between zero and one, representing a probability [3]. Figure 2.10 illustrates how different values all fall in the range.

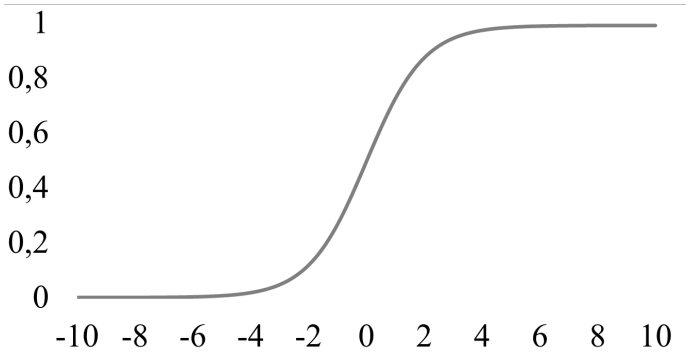


Figure 2.10: Plot of logistic function described in Equation 2.1.

The logistic function $f(x)$ is also described mathematically in Equation 2.1.

$$f(x) = \frac{1}{1 + e^{-x}} \tag{2.1}$$

To train the model the loss is calculated by the *maximum likelihood estimation*. This essentially maximizes the probability of a correct guess where any prediction higher than 0.5 is a guess of one and anything lower is a guess of zero [34].

Furthermore, the logistic regression function can be expanded for multiclass classification. To do this a technique called *one-vs-rest* can be employed [3]. One-vs-rest splits the problem into multiple binary classification problems where each class is compared to all the rest of the classes. This is done by creating a set of feature weights, essentially a unique model, for each class. These are then tuned to predict whether a given data point belongs to the class in question or any of the rest. The final prediction is the class that has the highest predicted probability of being true.

2.3.5 Ensembles

A single machine learning model can have glaring weaknesses. Furthermore, different models can have different weaknesses. To combat weaknesses but to utilize that different models can have different weaknesses one can assemble multiple models. This is done in hopes that a majority of the models will have a correct prediction and individual mistakes can be avoided. Arranging multiple models in such a way is called an ensemble. To make an ensemble you train multiple models on the same dataset and combine the models to make a single superior one [20]. In Figure 2.11 one can see how multiple models all train on the same data to then combine each of the models' predictions. There are several ways one can combine the models' predictions. Two of the most common ways are hard voting and soft voting.

2.3.5.1 Hard Voting

One of the most common ways to combine models in using hard voting. In hard voting, each model in the ensemble gets one vote. After voting the prediction with the most amount of votes becomes the ensemble's unified vote [20]. This is the example illustrated in Figure 2.11. An example of an ensemble that uses hard voting is random forests [30]. A random forest consists of a set of decision trees each trained on different subsets of the data. Once each tree is trained the forest makes a prediction based on the trees' votes [17]. One of the downsides of hard voting is that it disregards some information. Some machine learning models allow for the prediction of probabilities. To utilize this and weigh each vote one can employ soft voting.

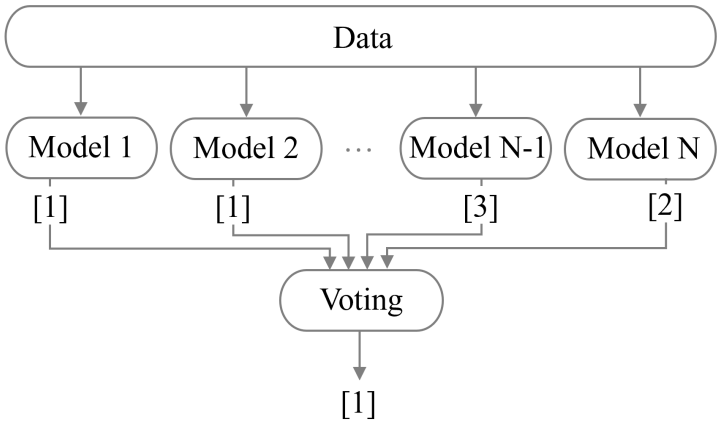


Figure 2.11: Ensemble using hard voting to aggregate the predictions of multiple models.

2.3.5.2 Soft Voting

Instead of simply having each model in an ensemble vote on a prediction one can weight each prediction. First of all, if a model is not confident in a prediction one should probably listen to that prediction less. Furthermore, each model can return confidence levels on the alternatives outside its first choice. For example, imagine a model is 60% confident in alternative one, 35% confident in alternative two, and 5% in alternative three. Using hard voting the vote would be cast on alternative one. But it is still relevant that the model prefers alternative two over alternative three. To utilize this we employ soft voting. In soft voting, we can simply sum each of the models' prediction confidence for each guess. The ensemble then predicts the alternative which has the highest sum [4]. This example is illustrated in Figure 2.12.

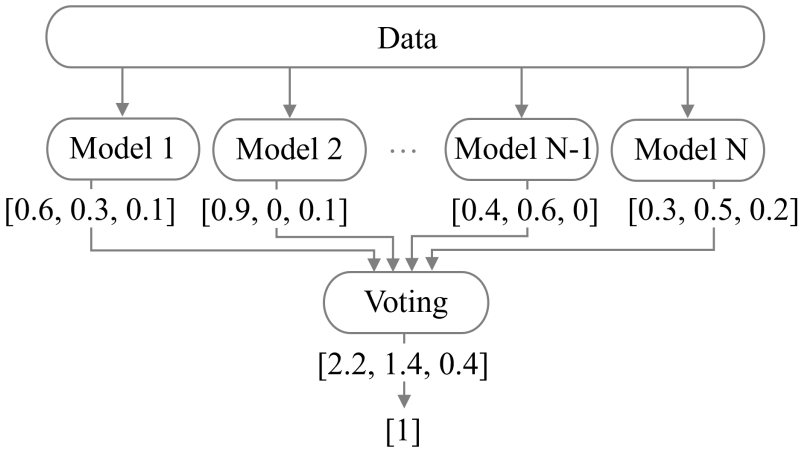


Figure 2.12: Ensemble using soft voting to aggregate the predictions of multiple models.

2.4 Evaluation

Probably the most fundamental metric in machine learning is accuracy. Accuracy is the percentage of correct guesses. It is useful since we want to achieve the maximum amount of correct guesses. Mathematically accuracy along with other metrics can be explained with the help of true positives, false positives, true negatives, and false negatives. Each one of these is illustrated by the confusion matrix in Figure 2.13 where the row represents a model's prediction and the column represents the truth.

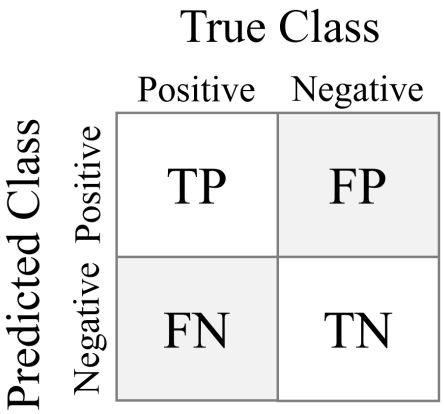


Figure 2.13: Confusion matrix of prediction and true classes where *TP* means *true positive*, *FN* means *false negative*, etc.

With the help of Figure 2.13 Accuracy can be explained by Equation 2.2.

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FN} + \text{FP}} \tag{2.2}$$

Although easily interpretable, sometimes accuracy can be misleading leaving room for other metrics. One such example is the $F1_{score}$ [38]. $F1_{score}$ is one of the most common metrics used in evaluating classifiers. The accuracy score is easier to understand but has flaws. Imagine classifying a rare disease where 99% of people are free from the disease. A model classifying everyone as healthy yields an accuracy of 99% [38]. Employing further metrics illustrates the flawed approach. The recall, how many positive cases were correctly classified, is 0%. We never correctly guess the state of a patient that has the disease. Precision is another metric that lands at 0%. Precision is the number of positive guesses that were correct. We never guess that a case was positive, the metric thus lands at 0%. To combat the flaws of accuracy we can utilize precision and recall. They are both useful so instead of using one of them, we can combine the two into their harmonic mean giving us the $F1_{score}$ [41]. The formulas of precision, recall, and $F1_{score}$ are shown in equations 2.3, 2.4, and 2.5.

$$\text{Precision} = \frac{\text{TP}}{\text{FP} + \text{TP}} \tag{2.3}$$

$$\text{Recall} = \frac{\text{TP}}{\text{FN} + \text{TP}} \tag{2.4}$$

$$F1_{score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{2.5}$$

These formulas hold strong in binary classification but as soon as there are more than two categories a problem arises. Our equations no longer work since you no longer guess positive or negative. With multiple categories, there are many guesses one can make. An easy way to solve this is simply to calculate precision, recall, and $F1_{score}$ with regard to whether or not you guess one of the classes correctly. After calculating an $F1_{score}$ for each class individually you simply take the average score [41]. This is called macro $F1_{score}$. There is another approached called micro $F1_{score}$. With this approach, we sum each true positive, false positive, and false negative to calculate a final micro $F1_{score}$ in the end. In the case of multiclass classification this befittingly leaves us with the percentage of correct guesses, accuracy in other words.

2.5 Previous Work

Problems regarding patent classifications have been previously tackled in the form of classifying IPC labels. IPC labels are labels that are manually assigned to patents. These are used to search through patents in a given technology domain. Since each patent can have multiple labels it is a problem for multi-label classification [23].

DeepPatent, an algorithm, consists of a convolutional neural network that is based on the abstract and title, labels patents' IPC sub-classes [23]. The authors further contributed with a patent dataset called USPTO-2M. DeepPatent uses supervised learning to train on the dataset. The dataset contains roughly 2 million patents filed in the United States Patent and Trademark along with their manual classification given to them in filing [23].

DeepPatent is outperformed on this dataset by PatentBERT [21]. PatentBERT is another contribution to the field and consists of a *BERT for patents* model fine-tuned on classifying IPC sub-classes. The model can with the help of extensive pre-training and the superior BERT architecture achieve state-of-the-art performance.

3

Data

For testing, Konsert provide a dataset. The dataset was created by manually building a technology tree and assigning patents to each category within the tree to create a patent landscape within material technology. The data is saved in an excel sheet with patents IDs and their respective category. As the technology trees can contain sensitive information some categories are not named. In general, the dataset contains patents regarding a material. Figure 3.1 illustrates the tree with the number of patents represented in each node. The first branching of the tree represents manufacturing on the left and application on the right of the material in question. Examples of applications (starting from the left in the right branch in Figure 3.1) include energy storage, high-speed electronics, biosensors, optoelectronics, composite coatings, and composites. The energy storage category includes the two categories batteries, and transistors and fuel cells. The dataset contains 6057 patents divided into 34 categories. Each category belongs to a parent category higher up in the tree. Figure 3.1 illustrates the hierarchical nature of the dataset with the number in each node representing the number of patents in that category. The Gray nodes with gray text are parent nodes with the number of patents being the sum of patents in its child nodes. Furthermore, the tree is divided into 3 levels. The lower the level the more granular the categories become. Level 3 is the most abstract category while level 1 is the most granular. Furthermore, each category in level 1 is a leaf node.

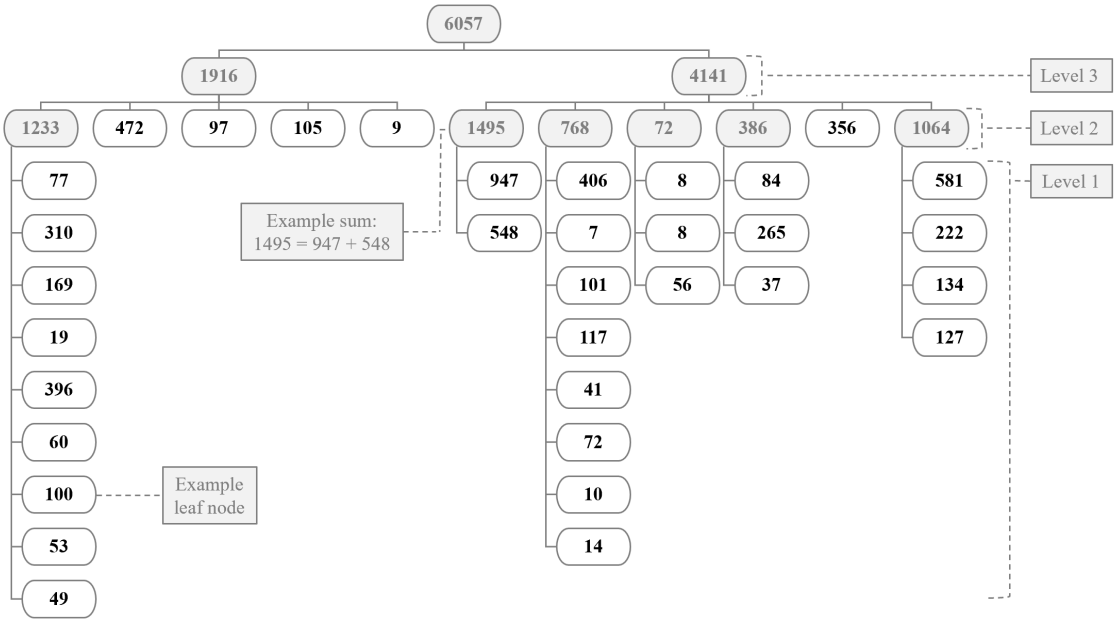


Figure 3.1: Tree describing the datasets structure. The leaf nodes with black text represent the number of patents in each class. The gray nodes represent the sum of the patents in its child nodes. Each level of the tree is marked to allow for reference in testing.

To classify these categories, more data about the patents than their ID is needed. To improve the dataset each patent's text and CPC classification will be added through web scraping.

3.1 Web Scraping and Pre-processing

Since more data than the patent's ID is needed for classification, the patent's text and CPC labels are web scraped from Google Patents. Google Patents is a service Google provides where one can find patents from patent offices such as the United States Patent and Trademark Office and the European Patent Office along with most major patent offices [31]. Using the service you can find most information about a patent such as a title, abstract, description, claims, and CPC classifications all in a standardized way. This allows for web scraping. Web scraping is a method where one collects information from websites and saves it in a standardized way by reading the website and extracting useful information.

There are several aspects of a patent that can be used to aid classification with the text probably being the most important. The abstract often contains a general overview but sometimes excludes vital details needed for some classification tasks. The patent description often contains some details excluded in an abstract. Both the abstract and description are thus web scraped. Since most BERT models cannot process more than 512 words the length of the text data is limited to that size. The text consists of the abstract and is then filled with the start of the description text until the size of 512 words is reached along with the abstract and description

tokens used in BERT for patents. Furthermore, the CPC labels are also saved. the classifications consist of strings, for example, *H01L27/3262*.

As the contents on a website consistently vary, collecting the abstract, description, and CPC labels sometimes fail. Any patents that have an abstract or description with fewer characters than 100 are removed. The average abstract and description length is far greater so these are considered to be outliers that cannot be used for classification.

In practice, Konsert would not be using a test set but to be able to compare our model a test set is created. This consists of 500 patents taken so that the class balance is the same as before. After removing the mentioned documents from the original 6057 we are left with 4760 patents and 33 categories instead of 34. 500 patents are then saved to a test set leaving us with 4260 patents organized in the way illustrated by Figure 3.2.

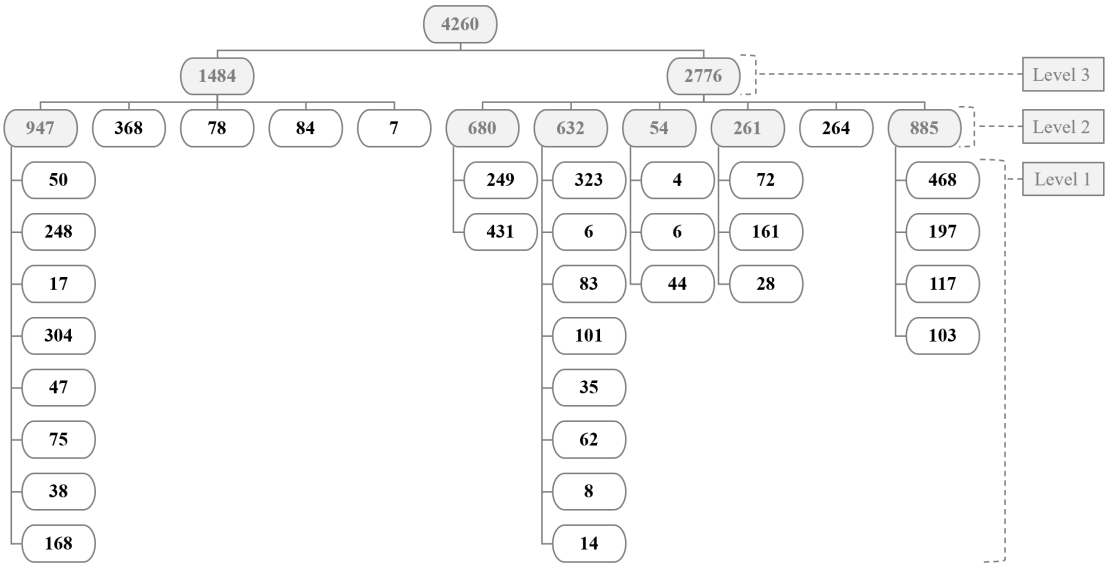


Figure 3.2: Tree describing the test sets structure after removing patents with short abstract and descriptions as well as separated a test set. The leaf nodes with black text represents the number of patents in each class. The gray nodes is the sum of the patents in its child nodes. Each level of the tree is marked to allow for reference in testing.

4

Method

To utilize the collected patent texts a few different text classification methods are employed. Several approaches are tested to explore what supervision level is optimal to classify patents with minimal manual work. Our lowest level of supervision is weakly supervised learning. For this approach, we employ a method called LOTClass made by Meng et al. (2020), which only requires keywords for each class [28]. For the next level of supervision, semi-supervised learning is employed. The chosen semi-supervised method is called MixText, which is made by Eva Sharma, Chen Li, and Lu Wang (2020) [9]. A method that only requires a small amount of labeled data (we only use 5-20 examples per class). This also allows us to test how the models' performance change with different amounts of labeled data. Finally, we propose a supervised approach we call LabelLR. LabelLR only requires CPC labels which might be quicker to learn than long texts. Because of this the supervised learning method is tested with the same amount of labeled data as MixText but without all the unlabeled data.

To achieve maximum performance a combination of the models is also tested. They are assembled into an ensemble to investigate what performance is possible if all the approaches work together. LOTCLass requires keywords and MixText and LabelLR always receive the same labeled documents in each experiment. The models' predictions are combined with the help of soft voting. If one of the models fails to run on a part of the data it will be excluded from the ensemble in that case. Each approach is further explained below along with what changes have been made.

These methods are compared to a fine-tuned BERT model trained on all available data. Each supervision level is illustrated in Table 4.1 along with what data they receive.

Supervision Level	Method	Data
Weakly supervised learning	LOTClass	keywords + unlabeled patents
Semi-supervised learning	MixText	5 labels/class + unlabeled patents 10 labels/class + unlabeled patents 20 labels/class + unlabeled patents
Supervised learning	LabelLR	5 labels/class 10 labels/class 20 labels/class
Supervised learning	FT BERT	Full dataset

Table 4.1: The different supervision levels that are tested from low to high.

4.1 Weakly Supervised Learning - LOTClass

LOTClass, made by Meng et al. (2020), is the chosen weakly supervised model [28]. To train, it requires only keywords along with text which is illustrated in Figure 4.1. But it is limited by how many texts it can find containing the keywords in question. The approach can be divided into 4 parts. Generating a category vocabulary, finding category indicative words, fine-tuning for masked category prediction, and finally self-training BERT on examples. A *BERT for patents* is the BERT model used throughout the method.

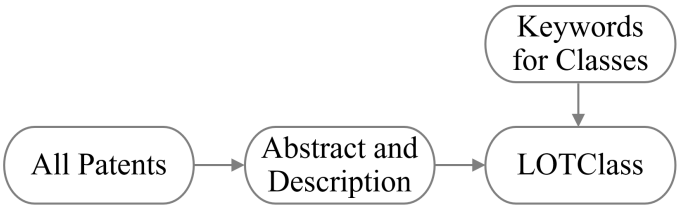


Figure 4.1: LOTClass requires only keywords along with text from the patent in training.

4.1.1 Generating Category Vocabularies

For a given category defined by a keyword, there are other words also indicative of that category. This bigger vocabulary of words can be used to give a broader understanding of the category instead of only using a single keyword. For example, the category and keyword ‘dog’ has words such as ‘Golden Retriever’, ‘pet’, and ‘animal’ that can all be used in similar contexts. We create such a vocabulary of similar words for each category following the method described below.

Words used in similar contexts are usually words with similar meanings. To find similar words to our keywords a BERT Masked language model (MLM) is used. The corpus is iterated over and each keyword occurrence is masked and fed into the MLM. Masking means that the word is hidden from the model. Based on the

surrounding words, the context, our MLM returns a probability distribution over each word in the vocabulary describing which word it thinks is most probable to be behind the mask. The top 50 most probable words are saved for each keyword occurrence. These are then compiled into a category vocabulary containing the 100 most commonly predicted words for each category. For example, a category named sport will get a vocabulary of 100 words which BERT finds are the most often interchangeable with the word sport [28]. How a keyword is masked and possible replacement words are generated is illustrated in Figure 4.2

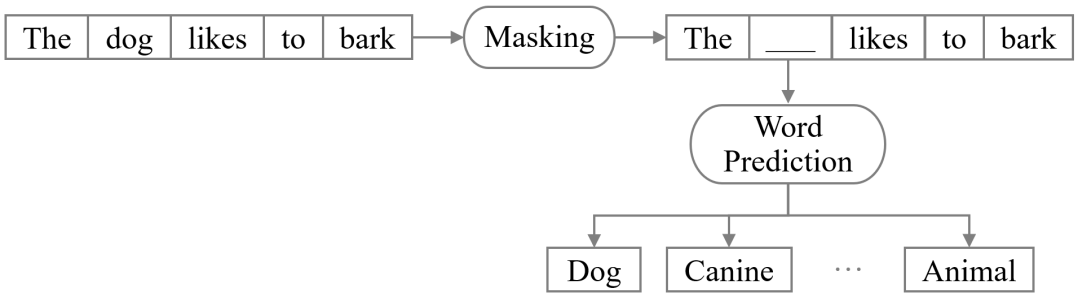


Figure 4.2: Example of how keywords are masked and a vocabulary is generated by predicting what words fit into the context.

4.1.2 Finding Category Indicative Word Occurrences

Based on each keyword in the generated category vocabulary the model can go through the corpora again and mask each keyword and check if the occurrence is category indicative or not. This is done by checking if the MLM predicts that a keyword occurrence can be replaced with similar words to the category vocabulary. Let's say we have a dog category and the dog breed 'boxer' is included in the category vocabulary. If, for example, the word 'boxer' is used as the dog breed boxer predicted words might include 'dog', 'animal', and 'canine'. If that is the case we would consider the keyword appearance category indicative. An example of such a comparison is illustrated in Figure 4.3. But if instead the word is used in the context of someone boxing other words might be predicted. Let's say words such as 'fighter', 'athlete', and 'sport' are predicted. In that case, the keyword is not considered to be category indicative of the category dog. In this implementation, a masked keyword is considered category indicative if 20 of the top 50 predicted words from the MLM are included in that category's 100-word category vocabulary [28]. This leaves us with a few labeled texts we can use in training for the next step.

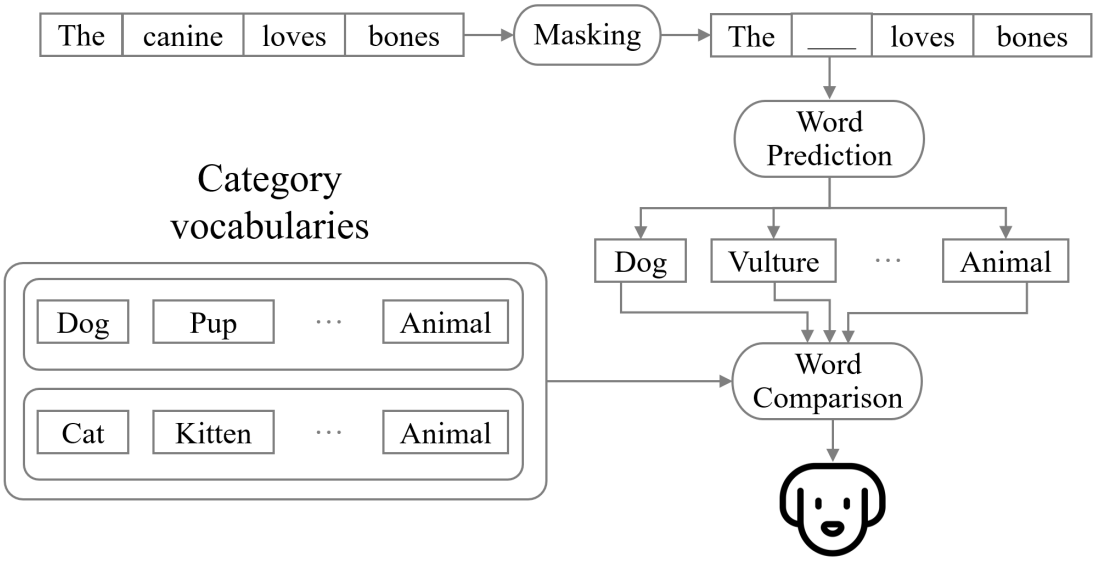


Figure 4.3: Example of how word predictions are compared with the category vocabularies to determine whether or not the occurrence is category indicative.

4.1.3 Fine-tuning for Masked Category Prediction

The above method for classifying what keywords can be used to create data set of instances with category indicative words. These are then used to fine-tune in a task called Masked Category Prediction (MCP) [28]. Instead of predicting what words can replace a masked word, our model is trained to predict what category the word belongs to directly, as illustrated in Figure 4.4. This is done for two reasons. Firstly, it teaches our BERT model to look for indications of our category in text. Secondly, it might be able to find more category indicative instances of the keywords.

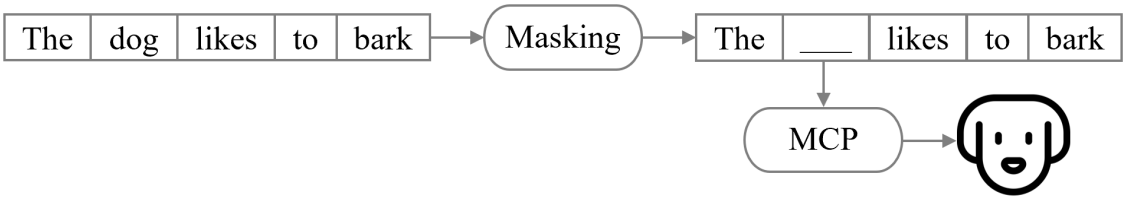


Figure 4.4: MCP consists of guessing a category based on context by a masked word.

4.1.4 Self-training BERT on Examples

After the MCP model is trained to predict what category a masked word belongs to we can now create a model capable of classifying any document regardless if there are category keywords present or not. Based on the occurrence of category indicative words we label a few documents. These can then be used to further fine-tune the BERT model. This is done for two reasons. Firstly, the keywords for each category are only found in some of the texts making a large chunk of the corpus unseen. Secondly, the MCP model masks words to predict what category they belong to,

meaning it removes information. We can instead fine-tune our model on the entire text. This leaves us with the final model capable of categorizing our dataset of unlabeled patent texts [28]. Bert predicting text without further steps is illustrated by Figure 4.5.

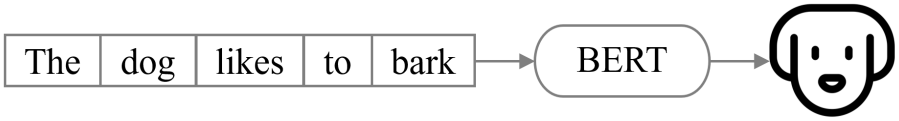


Figure 4.5: The BERT model can eventually predict categories based on the plain texts.

4.2 Semi-Supervised Learning - MixText

At a slightly higher supervision level, MixText is the semi-supervised method made by Eva Sharma, Chen Li, and Lu Wang (2020) [36]. MixText trains using a small amount of labeled data, around five to twenty labels per class, but utilizes plentiful unlabeled data. The data required to train MixText is shown in Figure 4.6. Just as LOTClass, MixText fine-tunes a BERT model. To adapt the approach to patents we utilize *BERT for patents* in MixText as well. One of MixText's techniques is to generate more data using back-translation. In back-translation, the text is translated into another language and then back to generate a slightly different text of the same label. This is not used to limit our scope.

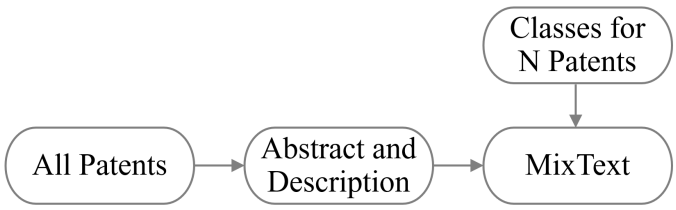


Figure 4.6: MixText takes patents texts, a few of them labeled with their classes to train.

4.2.1 Linear Interpolation

The first part of the process is a data augmentation technique called TMix [9]. The approach is based on a technique called Mixup. Mixup is data augmentation used for images where new data points are created in between two already labeled ones [50]. Imagine we have two data points with the same label. A given data point between the two probably has the same label. Furthermore, a data point between two other data points with different labels could be given a mix of the two labels. This concept is illustrated in Figure 4.7

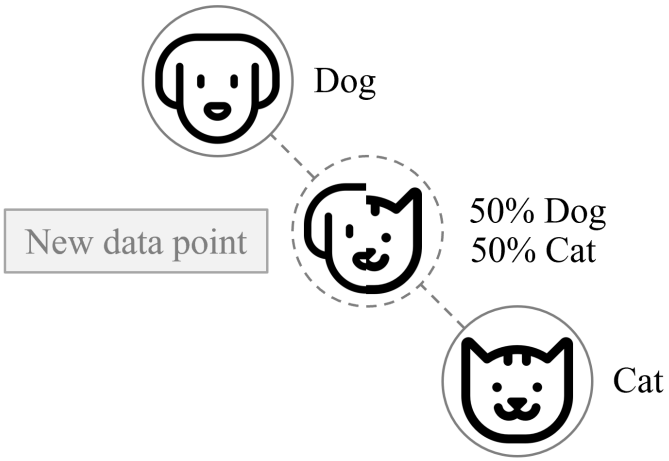


Figure 4.7: Linear interpolation between to data points and generation a new one that is a mix of the two original ones.

Mathematically it can be described in the following way. Given two data points along with their respective labels (x_i, y_i) and (x_j, y_j). We can get a new point in between x_i and x_j and a label in between y_i and y_j using linear interpolation as seen in Equation 4.1 and Equation 4.2 given that $\lambda \in [0, 1]$ sampled from a beta distribution [9]. This gives us more data points as well as forces the model's predictions to change linearly in between data points.

$$\hat{x} = \lambda x_i + (1 - \lambda) x_j \tag{4.1}$$

$$\hat{y} = \lambda y_i + (1 - \lambda) y_j \tag{4.2}$$

TMix augments image data but we want to apply the technique to text. This leaves us with a problem, how do we interpolate between texts. To do this we embed the texts using BERT for patents and interpolate in one of the hidden layers randomly chosen each batch between layers 7, 9, and 12.

4.2.2 Semi-supervision

To incorporate all the unlabeled texts we can utilize linear interpolation. To give the unlabeled document a label we have our model predict their labels. The two sets of now labeled text are combined into a super set. This set is then used by choosing two points randomly and performing the linear interpolation [9]. An example of how new data points can be generated from classified unlabeled data points can be seen in Figure 4.8.

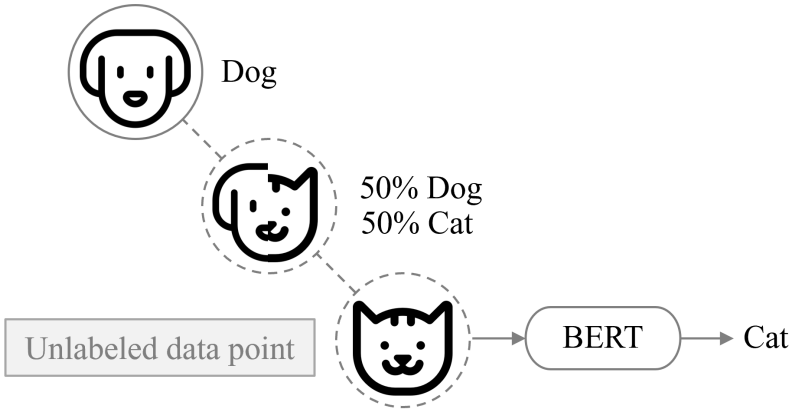


Figure 4.8: New data points generated from classified unlabeled data points.

4.2.3 Entropy Minimization

Entropy minimization is also employed on the unlabeled text. This is done to get predictions with higher confidence and minimize entropy within the prediction. Practically this is done by adding a loss to the existing TMix Loss. With γ being a tunable hyperparameter, x_u being an embedded unlabeled text, and y_u being a label prediction for that text the added loss can be described by Equation 4.3.

$$L_{margin} = max(0, \gamma - ||y_u||_2^2) \tag{4.3}$$

This leaves us with a final loss for the unlabeled text with L_{TMix} being KL-divergence between the predicted and the interpolated label for a given datapoint:

$$L_{MixText} = L_{TMix} + \gamma_m L_{margin} \tag{4.4}$$

4.3 Supervised Learning - LabelLR

Although patents' texts are useful they also contain CPC labels. We propose an approach we call LabelLR to use the CPC labels to classify our dataset's classes. The method uses supervised learning but since CPC labels are less complex than text no more data than for MixText is required (5-20 labeled documents per class). LabelLR only requires the CPC labels and the classes of a few patents as shown in 4.9 to train.

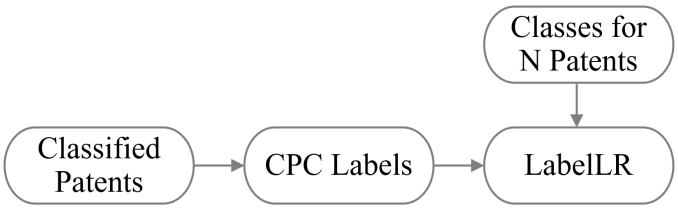


Figure 4.9: LabelLR requires a few classified documents along with their CPC labels to train.

CPC labels are hierarchical. To utilize this we extract the classifications section, class, sub-class, group, and sub-group. Meaning if we have the label *H01L27/3262* we also add the labels *H*, *H01*, *H01L*, and *H01L27*. This step is done for each label. Two labels may be part of the same group. Let's say several labels are in the *H* section. In this case, duplicate *H* labels are removed. To train the model one feature is created for each possible label and is annotated 1 or 0 for each patent depending on if they contain that label. These features are then fed into a logistic regression model. For prediction, the labels contained in the patent can be fed to the trained logistic regression model as illustrated in Figure 4.10.

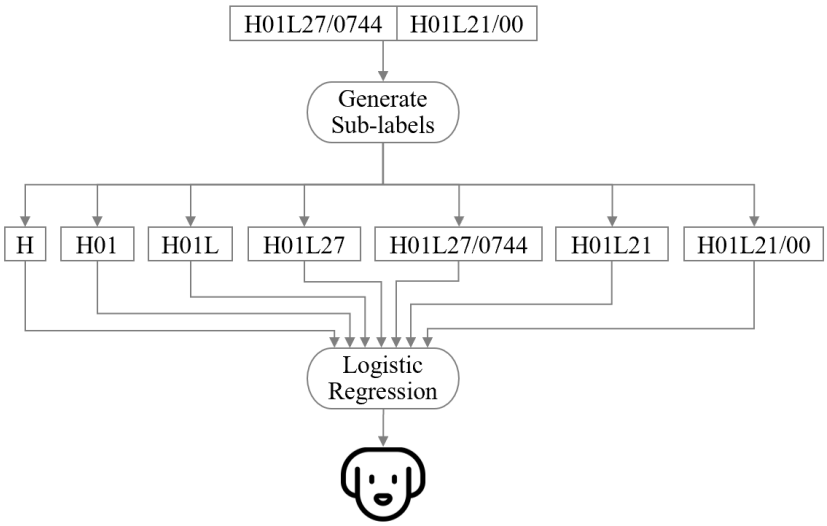


Figure 4.10: CPC labels are split into their sub-parts and fed to a logistic regression model in LabelLR.

4.4 Ensemble

To test whether or not the models perform better together they are assembled into an ensemble. This is done by letting MixText and LabelLR train on the same 10 labeled documents per class and LOTClass train using keywords. Each model then gives its predictions on the test set. The different predictions are then united into one prediction through soft voting. Figure 4.11 illustrates how LOTClass requires keywords and patents text while MixText only requires the text and LabelLR uses the CPC labels to make predictions.

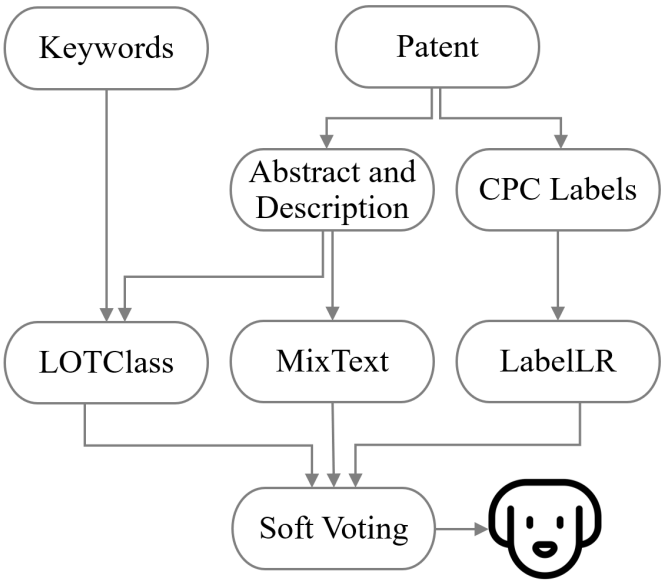


Figure 4.11: Our different methods use different data and are combined using soft voting to predict a category.

4.5 Supervised Learning - Fine-tuned BERT

To understand how well the different models perform, a supervised approach trained on the whole text dataset is also used. This is implemented to understand what is possible using the dataset. To do this the parameters of a BERT for patents model are fine-tuned on our dataset. Practically this is done by adding a layer of neurons to the output of the model with each added neuron corresponding to a category. Similar to our other approaches the difference is that in this approach we have access to all the data labeled. This thus sets a benchmark of what is possible.

4.6 Evaluation

There are several ways of classifying categories in a tree structure. Firstly, we will simply regard each leaf node as a category and then move on with normal classification directly yielding a result. Our second approach is to classify the patents starting from the top of the tree and regard each branch as a new classification task. Classifying in this way iteratively divides the patents into more and more fine-grained classes eventually reaching the leaf nodes. The branches can be divided into levels depending on how far down from the root node they are. These levels are illustrated in Figure 3.2 as level 1, level 2, and level 3. Using such an approach does require doing more classifications and thus training several models. It does however come with a benefit. It allows us to evaluate how well our methods perform with categories at different levels of granularity.

To evaluate how well each of the models performs we utilize the macro and micro $F1_{score}$. A model is trained and the scores are calculated for each parent node

of the tree as well as for all the leaf nodes. Since there are 9 parent nodes 9 models will be trained plus 1 for all leaf nodes resulting in a total of 10 models for each experiment. In the case of the ensemble if any of the individual models that cannot classify a certain level of the tree will be excluded from the test. Further tests are run with 5 and 20 labeled patents on the leaf nodes and the top level, level 3. This is done to allow for analysis of how a different data amount can change the results without having to train another 10 models for each data amount.

4.7 Experimental Setup

The setup used to run the methods is Google's Colaborary (Colab). Colab is an online code environment that gives users access to cloud-based computing. This makes Colab good for collaboration and allows for easy access to good computing resources including GPUs [26].

For this project, Colab Pro+ is used to get access to longer run times, more RAM, and better GPUs. The GPUs available in Colab Pro+ vary depending on which ones are available at the time [26]. Examples of GPUs we had access to include an 'NVIDIA Tesla P100-PCIe-16GB' and an 'NVIDIA Tesla V100-SXM2-16GB'.

Using this setup five different experiments are run. They all are trained and evaluated on the same train and test data. All models are trained and evaluated on each level of the hierarchical dataset as well as all the individual leaf nodes.

4.7.1 Experiment 1 - LOTClass

The first experiment consists of training LOTClass on the training data. The hyperparameters are matched with the parameters used in LOTClass's paper [28]. Some parameters were changed to allow for our less powerful system to run the model. The keywords used as well as the category vocabularies generated by LOTClass are shown in Appendix A.1 and Appendix A.2 for the tests that run successfully.

The BERT model used for LOTClass is BERT for patents [33]. The maximum amount of tokens for each document is set to 256 tokens. The training batch size is 128 with 5 epochs run. The optimizer used is Adam and the learning rate is 0.00002 for MCP and 0.000001 for self-training. The keywords for each class are chosen by writing down the name of the class and its subclasses.

4.7.2 Experiment 2 - MixText

For the second experiment, a MixText model is fitted to the training data. The hyperparameters are chosen from mainly two factors: the default parameters used on MixText's GitHub [8], and what works with our available computer resources.

The MixText method is given 10 labeled documents per class, with the rest being unlabeled. Two epochs are run using 1000 validation iterations each epoch. Where

validation iterations are the number of documents the method manually classifies from the unlabeled data each epoch. The learning rates are set to the same as in MixText's GitHub [8]. An initial learning rate for BERT at 0.000005 and an initial learning rate for the whole model at 0.0005. The maximum amount of tokens from each document the model has access to is the first 256. The α in the beta distribution is set to 16, and the temperature T for the sharpen function is set to 0.5. Back translation is not used for data augmentation. The batch size for both the labeled and unlabeled is set to 1. The BERT model used is BERT for patents [33].

4.7.3 Experiment 3 - LabelLR

LabelLR's logistic regression model uses the default hyperparameters from Scikit-learn [35]. It recieves the same 10 labeled documets per class as MixText.

4.7.4 Experiment 4 - Ensamble

The fourth experiment consists of testing an ensemble of the three models using soft voting. LOTClass is excluded from the ensemble when it fails to run.

4.7.5 Experiment 5 - Fine-tuned BERT

For the fifth experiment, a *BERT for patents* model is fine-tuned on all the training data and is run for 5 epochs. The maximum token length of the texts is 256 tokens and an Adam optimizer is used with the learning rate set to 0.00002.

5

Result

This chapter contains the results of the experiments in two forms. Firstly, each approach's results will be described separately in a tree following the dataset's structure illustrated in Figure 3.2. Each node contains the macro and micro $F1_{score}$. For example, if a node says *70.3/75.5* it means that the macro $F1_{score}$ is 70.3 and the micro $F1_{score}$ (accuracy) is 75.5. The exception is the leaf nodes that instead show how many classes it represents. This way of showing the $F1_{score}$ is illustrated by Figure 5.1.

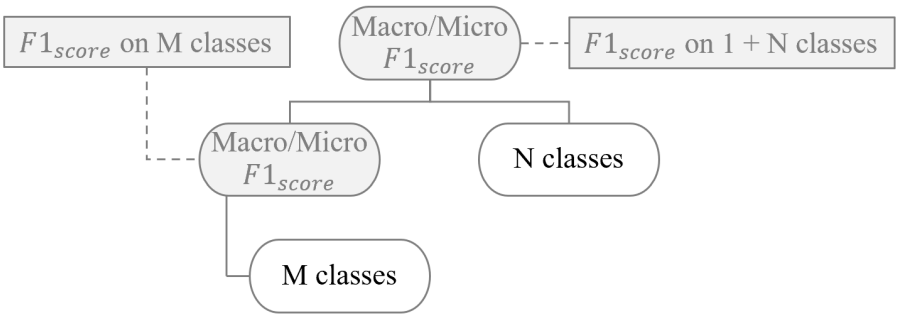


Figure 5.1: Tree describing how the results are displayed.

The data tree is divided into 3 levels illustrated by Figure 3.2. The average macro and micro $F1_{score}$ is summarised for each level and each method in Table 5.1. The second column shows what data is used by the methods. For example "10/class" means that the model is given 10 labeled patents per class. Both MixText and LOTClass additionally received all the unlabeled patent texts. For some levels, tests are conducted on 5 and 20 labeled documents which can be seen in the table. The final row shows the results for a fine-tuned BERT (written as FT BERT) trained on a fully labeled dataset. For each level, the highest score is in **bold** excluding the results for the fine-tuned BERT.

Method	Data	Level 1		Level 2		Level 3		Leaf Nodes	
		F1		F1		F1		F1	
		Macro	Micro	Macro	Micro	Macro	Micro	Macro	Micro
LOTClass	**Keywords**	60.7	64.2	–	–	38.9	42.4	–	–
	5/class	72.0	77.4					44.9	54.0
MixText	**10/class**	80.1	81.6	**70.7**	68.5	60.4	73.3	51.4	60.7
	20/class	**85.6**	**87.2**					**55.3**	**64.1**
	5/class	55.7	55.8					18.1	24.6
LabelLR	**10/class**	64.0	64.2	46.6	49.4	45.2	54.6	25.2	31.5
	20/class	66.2	66.7					26.5	34.5
	5/class	72.4	76.7					42.2	53.6
Ensemble	**10/class**	79.9	81.3	70.25	**71.75**	**62.4**	**77.1**	50.3	61.8
	20/class	83.5	85.1					51.1	62.0
FT BERT	**Full dataset**	87.5	89.1	84.6	88.1	69.6	84.4	61.3	71.4

Table 5.1: The average macro and micro $F1_{score}$ for each level of the dataset including performance on all leaf nodes. Blank cells are tests that were not conducted while cells marked with '–' failed. The best score for each level is shown in **bold** excluding the results for the fine-tuned BERT (FT BERT in table).

Observing Table 5.1 we can see that LOTClass yields the lowest scores with tests on the leaf nodes failing. MixText has the highest scores of all the individual approaches with a higher number of labeled documents improving performance. A notable difference is that the ensemble tends to perform better on lower levels, not including the test on the leaf nodes. None of the approaches can beat the fine-tuned BERT models shown in the last row. Although with 20 labeled documents per class the ensemble and MixText are not far off.

In Figure 5.2 the results for LOTClass is shown. Figure 5.3 and 5.4 shows the results of MixText and LablLR both trained on 10 labels per class. The results for the combined ensemble is shown in Figure 5.5. Finally the fine-tuned BERT model is shown in Figure 5.6.

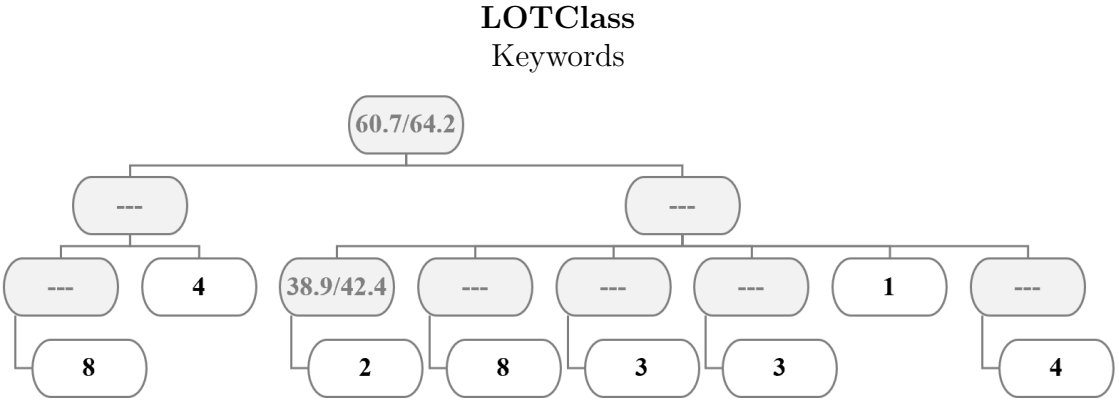


Figure 5.2: Tree following the datasets structure with each node describing the macro and micro $F1_{score}$ that LOTClass achieved on each branch. 3.2

Observing Figure 5.2 we can see that only two tests were completed by LOTClass which were application/manufacturing and the energy storage classes. This is caused by a lack of available patents in each class. Furthermore, the successful tests were both completed on branches where two classes were categorized.

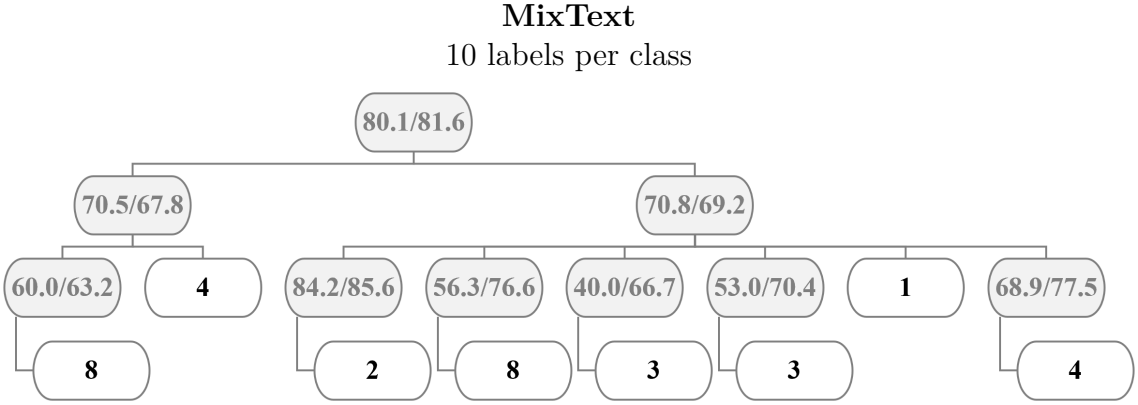


Figure 5.3: Tree following the dataset's structure with each node describing the macro and micro $F1_{score}$ that MixText with 10 labeled patents per class achieved on that each branch. Further description of the leaf nodes is described in Figure 3.2

MixText's results in Figure 5.3 show that the results are similar on the higher levels while on level 1 the scores vary more. For example on the third leaf node representing energy storage, the macro $F1_{score}$ is 84.2 while on other nodes such as biosensors on the third leaf node on the right branch goes down to a score of 40.0.

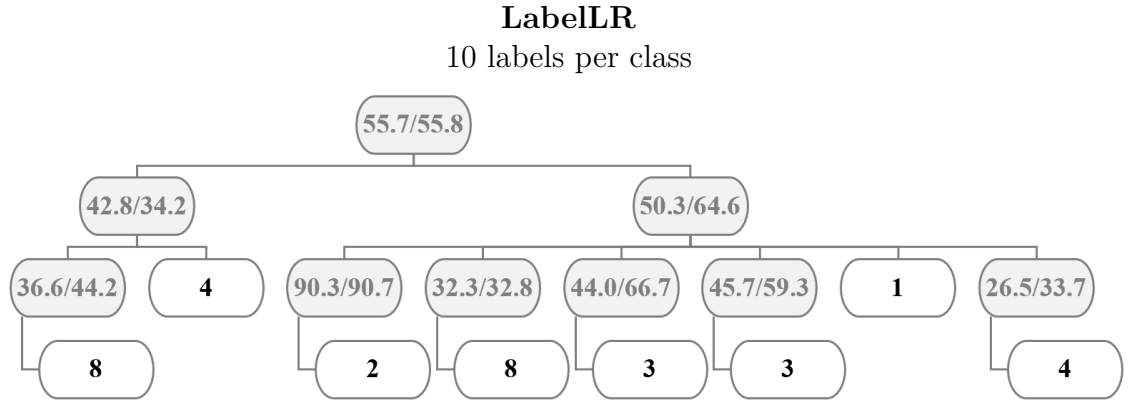


Figure 5.4: Tree following the dataset's structure with each node describing the macro and micro $F1_{score}$ that LabelLR with 10 labeled patents per class achieved on each branch.

Figure 5.4 shows the results for LabelLR which vary significantly. Notably, the best results yield a macro $F1_{score}$ of 90.3 on the class energy storage while the worst is 26.5 on composites.

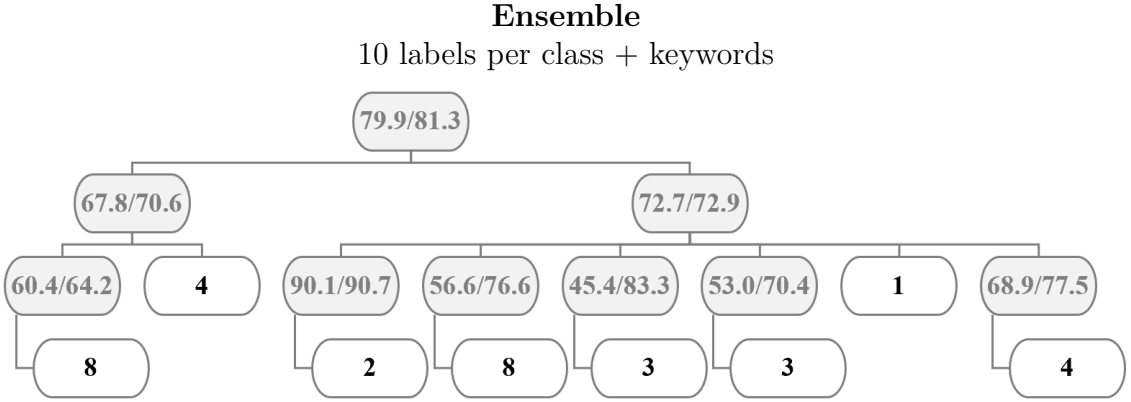


Figure 5.5: Tree following the dataset's structure with each node describing the macro and micro $F1_{score}$ that LabelLR with 10 labeled patents per class achieved on each branch.

The results in Figure 5.5 show similar results to MixText although some nodes have seen slight improvement matching where LabelLR performs well.

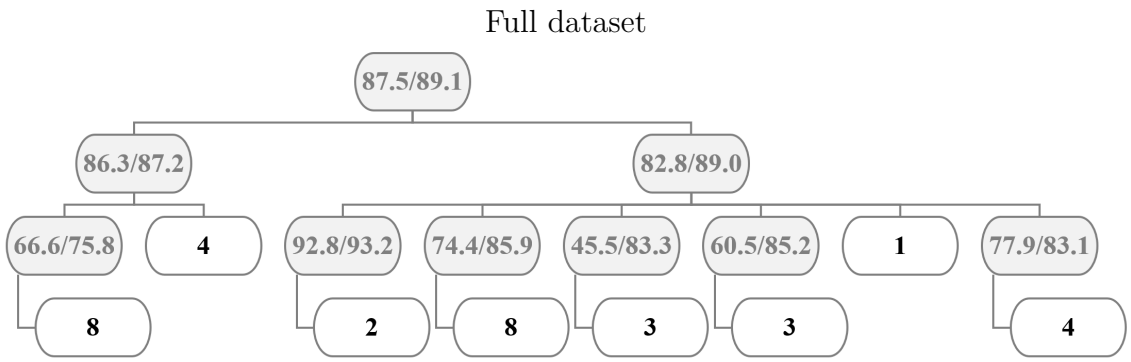


Figure 5.6: Tree following the datasets structure with each node describing the macro and micro $F1_{score}$ our fine-tuned BERT model can achieve

Finally, Figure 5.6 shows the results for our fine-tuned BERT which performs well on all tests.

6
Discussion

In this chapter, each model, the dataset, practical applicability, and possible future work is discussed.

6.1 Models

In this section, each one of the models' results are discussed.

6.1.1 LOTClass

The most notable thing about LOTClass is that it fails on the majority of tests. In other tests using LOTClass, big datasets are used with plenty of examples. Our dataset is more limited. Some classes contain only a handful of documents as seen in Figure 3.2. Using keywords to find these few examples is thus difficult. Even if the approach worked perfectly and all available documents in a small class were found, fine-tuning on these few examples would most likely yield disappointing results.

The approach fails on many of the small classes but how does it perform when it can find enough documents. Since LOTClass is keyword-based finding example documents in the high-level categories might be hard. Let's say you have a patent regarding dogs. A high-level category might be 'organism' and a low-level category might be 'dog'. 'organism' is unlikely to appear in the patent since it is too general while 'dog' is more likely to appear. This can be an explanation as to why the approach does not perform well on high-level classes. When you instead classify low-level categories there is a different problem. There are fewer and fewer patents in each class. This can cause the approach to fail just because it cannot find enough examples even though many might contain relevant keywords. Because of these two problems, LOTClass struggles with our dataset. Observing the results it did achieve we can also see that both tests only had two categories. This means that simply guessing the majority class would yield a micro $F1_{score}$ of over 50% which is higher than LOTClass got on one of the tests.

6.1.2 MixText

MixText is the approach that requires the most amount of manual annotation alongside LabelLR. It requires the user to manually label example documents but also yields the best results. Considering the failure of LOTClass to perform and its superior performance over LabelLR it is the best approach. MixText's results also

vary a lot. In Figure 5.3 we can see that on the top class with 10 labels per class it achieves a macro $F1_{score}$ of 80.1 while on a lower level it goes down to 40.0. A more granular class (the lower level classes) might require a better understanding of specific nuances to classify. On a high abstraction level, a general context might be good enough while on lower levels you have to understand specifics of a technology area that MixText cannot do.

On level 3 we can see that performance is heavily dependent on how many labeled documents are available. On the leaf nodes, this is not the case. One possible explanation is that the individual labeled documents don't represent the overall class very well. The top-level is manufacturing and application of a certain material technology. A lower-level application patent might be a concrete example. For MixText to understand that the category is application many patents might be required to understand the overall class. Furthermore, when you randomly take 10 labeled documents for the high-level classes the patents in question might be a bad representation of the class. The patents are randomly taken from sub-classes that may or may not represent the parent category in a good way. While if you are categorizing in a lower level the documents do not vary as much.

6.1.3 LabelLR

Using the CPC labels for classification generally didn't perform very well. Although with an exception. On battery storage, over 90 percent accuracy (micro $F1_{score}$) is reached. Generally, the model is not expected to perform well on its own but was instead meant to complement MixText in the ensemble. Considering this, achieving 90 percent accuracy is impressive. The reason why the model generally does not perform great is because the existing CPC labels are simply not enough to categorize into custom categories. And since LabelLR is not provided with more data better performance is hard to achieve. But in the case where it achieves over 90 percent accuracy, it is most likely the case that the existing CPC labels are close to the custom categories the dataset contains. In the example of energy storage the CPC subclass *H01M* represents batteries [32] making the battery patents easy to separate from the rest. On the other hand a CPC label for manufacturing of the material in general was not found. In situations like energy storage relying on the CPC labels for classification is a great option. The only problem is that they seem to be rare to come by.

There are also some cases of significantly higher micro than macro $F1_{score}$. This is caused by imbalanced classes. It can be that our LableLR finds that classifying a larger class often is beneficial but fails to perform as well on the smaller classes. Since macro $F1_{score}$ is a balanced metric this preference for larger classes will be reflected in the score, unlike micro $F1_{score}$.

6.1.4 Ensemble

For most of the tests, the ensemble only consisted of MixText and LabelLR since LOTClass failed. LabelLR had varied results and combining it with MixText could improve performance in several cases but not significantly. This could be because the different models probably failed to classify similar patents. It could be that any time LabelLR succeeded there was no problem for MixText to also do a correct classification. But in the rare case that LabelLR could excel the ensemble could take part as seen by the fact that the ensemble reaches score of over 90 percent in the same places as LabelLR.

6.2 Data

The dataset used stands at the core of many of the struggles. The smaller size of the datasets made the models struggle to learn. LOTClass could not even get enough data to self-train correctly on the smaller subclasses of the datasets. When LOTClass could train it was often with a small sample of category indicative documents. MixText on the other hand can run on all subclasses of the dataset even though it still suffered from a small number of unlabeled documents to train on. This smaller size of datasets also made the test set small and in several instances small enough that removing one patent from the test set could have a rather large impact on the evaluation. This can mean that the results have high variability depending on the patents chosen for the test set.

Only testing the methods on one dataset is also a downfall. Only testing on one reduces our ability to generalize results since this dataset might not be representative of the average dataset Konsert encounters.

Manually labeling a dataset in the real world is tiring work. This can lead to a consultant using shortcuts when it comes to labeling patents. In research, datasets can be labeled several times by several people to make sure to get rid of mistakes. This has not been done with our dataset which increases the probability for mistakes.

6.3 Practical applicability

Evaluating how good a method has to be to prove useful in practice for Konsert is difficult. One can argue that even a flawed method is useful. First of all, the classifications can be the beginning of a patent landscape construction process. Letting consultants manually classify the least confident predictions can refine the predictions enough to be used. Secondly, perfect predictions are not necessary. If you want to see general trends in a technology it is okay to have some patents with the wrong class. What level of incorrectly predicted patents is acceptable is up to Konsert to decide. This leaves us with another issue. There is no way to know how well a model performs on a new unlabeled dataset.

In real-world use, the process would also look a bit different. We randomly select labels that are given to our models. But in practice, it is done differently. Instead, the patents that best represent the class are chosen by consultants. In our case, the patents chosen could even be of the wrong class because of incorrect classifications on the dataset. By selecting the patent labels to be used carefully performance could improve.

Some practical aspects should be considered if Konsert wants to employ a machine learning based method for classifying patents. Firstly, it is impossible to know how well the model would perform on a new dataset. Because each new dataset has unique categories and patents it is impossible to guarantee a level of performance. Secondly, it requires knowledge to operate and understand the weaknesses of the method. This could mean that Konsert would need more machine learning expertise to successfully utilize the method. Thirdly, the models tested in this paper are far from practical for a regular consultant. In the models' current state large amounts of manual work needs to be performed before the predictions are made such as pre-processing the data, and modifying and running the code.

6.4 Future Work

Limited resources leave our exploration equally limited. Because of this, there are several opportunities for improvements that could be explored in the future.

6.4.1 Patents' Unique Characteristics

To boost performance patents' CPC classifications are used. Although useful, several other aspects can also be used. First of all the inventors. The inventors of a patent are probably experts in their field. Because of this, it is not unreasonable to expect that they will issue more patents in the same field. This can be utilized in patent classification. If an inventor has many patents in a certain technology chances are that another patent from the same inventor is in the same or a similar field. An inventor cannot alone determine the category of a patent, but it could certainly provide a boost to performance. This could be implemented using logistic regression in a similar way to how the CPC classifications are currently being used in LabelLR. A patent's inventor is only one of the many aspects that could be incorporated from a patent. A patent's geographical origin or the owner of a patent such as a company provides similar use.

More complex parts of the patents could also be used. All patent contains references to other patents. Patents referencing each other are probably in similar technologies. Utilizing this fact, one can find clusters based on how the patents are connected. A patent is also often part of a patent family. A patent family is a group of patents all protecting the same invention. But there can be different patents for different geographical areas. Although these aspects might not be as simple to integrate as the inventor of a patent. The question as to how it can improve patent classification remains open.

6.4.2 Back-translation

To limit the thesis back-translation is not used in MixText to augment data. From time to time the method struggled. Furthermore, some patents originated from countries such as China and have thus been translated. Augmenting the data by translating the patents to Chinese and back provides more data to improve performance. It also generates English patents with grammar more similar to the Chinese ones as they have now also been translated. The method is of course not limited to Chinese and any back-translation could have a significant impact on the performance of MixText.

6.4.3 Transfer-Learning

Transfer learning is a useful tool when data is limited. Training a model on another dataset to prepare it for the challenge ahead can prove useful. It could be used in Konsert's application. Patents are plentiful and most already have hand-labeled CPC classes. Fine-tuning any of the BERT models to first categorize groups within the CPC classes could prepare it for patent landscaping and might boost performance.

6.4.4 LOTClass Keywords

LOTClass can find valuable insights using keywords. This leaves the issue of finding good enough keywords. To boost the performance of the model one can provide more and better keywords. This can be done using external databases. For example, WordNet consists of relations between words [29]. This can be used to find synonyms expanding on our list of keywords that LOTClass can use to search through patents. Using external databases is a quick way to find more keywords but tools such as Word2Vec can also be used [10]. Word2Vec embeds words. These embeddings can be used to measure the similarity between words. Based on this the keyword vocabulary of LOTClass can be expanded.

6.4.5 Combining LOTClass and MixText

The models are incorporated into an ensemble. This is not the only way to combine the models. The technique used to find patents to fine-tune a BERT model in LOTClass could instead be fed into a MixText model. The LOTClass approach might not find many examples and MixText can thrive with little data. Exploring if this could create one better model is future work. Although such a model does come with a downside. When combining the models in an ensemble they hopefully work against each other's flaws. When combined into one, the benefits of being in an ensemble disappears and overfitting might become a bigger issue.

6.4.6 More Patent Datasets

Konsert's technology trees are custom-made for each client. This means that the data is not publicly available. Comparing our approach to others is thus difficult.

Future work can be to apply the approach to publicly available data. This would allow for comparison between more models. Furthermore, it would allow us to assess how difficult the categorization of Konsert's patents is compared to other datasets.

6.4.7 Hierarchical Classification

We have two approaches to classifying categories in a tree structure. Firstly, we categorized the leaf nodes. Secondly, we categorized each level of the tree. But there are more approaches. One approach would be to regard the problem as one of multi-label classification. To do this any patent would be labeled with its leaf node category as well as its parent classes. To would additionally mean that a patent does not have to belong to any leaf node and could instead only be categorized higher up the tree. How this would affect general performance remains to be seen.

6.4.8 Comparison to Search Engines

Currently, Konsert uses search engines to find patents such as *Google Patents*. Although, Google Patents does not provide custom categorization searching for category keywords probably retrieves relevant information. An interesting point to investigate could thus be to compare what type of patents Konsert receives with simple searches in search engines compared to our approach.

6.4.9 Variability in Results

Because which example patents are randomly selected for MixText has an impact on the performance the size of the impact could be measured. To measure this variability the MixText model could be trained multiple times each time with different random patents to get a range of results. From this, an average result and a standard deviation could be extracted to better understand the variability.

6.4.10 Explore Other Weakly and Semi-supervised Methods

The weakly supervised method, LOTClass, and the semi-supervised method, MixText, are not the only viable options for patent classification. To fully understand what weakly and semi-supervised approaches work best for patent classification other methods should be explored. Two examples of weakly supervised methods that would be interesting to test with patent classification are ClassKG [51] and X-Class [47] which both rely on keywords.

7
Conclusion

This thesis set out to explore weakly supervised, semi-supervised, and supervised learning for patent classification. Employing LOTClass, MixText, and LabelLR leads us to some conclusions.

Semi-supervised learning is the supervision level most fitting for classifying patents with minimal manual labor. The semi-supervised approach, MixText, performed best on most tests. Meanwhile, the weakly supervised approach, LOTClass, often fails to run due to a lack of available text leave keyword instances rare to come by.

Using a unique property of the patent, the CPC labels, LabelLR can classify patents with the help of few examples. Even though it uses supervised learning LabelLR can be trained using less than ten labeled examples per class. It performs best on some granular classes probably where the CPC labels fit well. This leaves the opportunity for LabelLR to be combined into an ensemble with MixText which is not as affected by class granularity.

When LabelLR is combined with MixText into an ensemble it performs similarly to MixText. But MixText alone seems to be the best alternative reaching an accuracy (micro $F1_{score}$) of 60.7% on 33 classes with ten labeled patents per class, not far from what a fine-tuned BERT model achieves. A great start for Konsert to serve their clients. Although, the path to a useful product for Konsert's consultants is long. It requires streamlining data pre-processing and the creation of an interface for the non-coder. But for now, this thesis shows that patent classification with low levels of supervision is feasible.

Bibliography

[1] Stephen Adams. *Information Sources in Patents*. De Gruyter Saur, 2020. ISBN: 9783110552263. DOI: 10.1515/9783110552263.

[2] Yoshua Bengio, Réjean Ducharme, and Pascal Vincent. "A neural probabilistic language model". In: *Advances in Neural Information Processing Systems* 13 (2000).

[3] Ekaba Bisong. "Logistic Regression". In: *Building Machine Learning and Deep Learning Models on Google Cloud Platform: A Comprehensive Guide for Beginners*. Berkeley, CA: Apress, 2019, pp. 243–250. ISBN: 978-1-4842-4470-8. DOI: 10.1007/978-1-4842-4470-8_20.

[4] J. Brownlee. *Ensemble Learning Algorithms With Python: Make Better Predictions with Bagging, Boosting, and Stacking*. Machine Learning Mastery, 2021.

[5] F Cariaggi, C De Nobili, and S Bratières. "Patents for Industrial Pollution Prevention and Control". In: (2021).

[6] Mauro Castelli, Leonardo Vanneschi, and Álvaro Rubio Largo. "Supervised learning: classification". In: *por Ranganathan, S., M. Grisbskov, K. Nakai y C. Schönbach* 1 (2018), pp. 342–349.

[7] H. Charmasson. *Patents, Copyrights and Trademarks For Dummies*. –For dummies. Wiley, 2004. ISBN: 9780764525513.

[8] Jiaao Chen and Diyi Yang. *MixText*. 2020. URL: https://github.com/GT-SALT/MixText/blob/master/README.md (visited on 04/25/2022).

[9] Jiaao Chen, Zichao Yang, and Diyi Yang. "MixText: Linguistically-Informed Interpolation of Hidden Space for Semi-Supervised Text Classification". In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. 2020, pp. 2147–2157.

[10] Kenneth Ward Church. "Word2Vec". In: *Natural Language Engineering* 23.1 (2017), pp. 155–162.

[11] P.D.S. Consulting. *Data Science for Business Professionals: A Practical Guide for Beginners (English Edition)*. BPB PUBN, 2020. ISBN: 9789389423280.

[12] Blaise Cronin. "Annual review of information science and technology". In: (2004).

[13] Pádraig Cunningham, Matthieu Cord, and Sarah Jane Delany. "Supervised learning". In: *Machine learning techniques for multimedia*. Springer, 2008, pp. 21–49.

[14] Abhijit Dasgupta et al. "Brief review of regression-based and machine learning methods in genetic epidemiology: the Genetic Analysis Workshop 17 experience". In: *Genetic epidemiology* 35.S1 (2011), S5–S11.

[15] Jacob Devlin et al. “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding”. In: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. Minneapolis, Minnesota: Association for Computational Linguistics, June 2019, pp. 4171–4186. DOI: 10.18653/v1/N19-1423. URL: https://aclanthology.org/N19-1423.

[16] Jacob Devlin et al. “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding”. In: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. 2019, pp. 4171–4186.

[17] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer Series in Statistics. Springer New York, 2013. ISBN: 9780387216065.

[18] Kamran Khan et al. “DBSCAN: Past, present and future”. In: *The fifth international conference on the applications of digital information and web technologies (ICADIWT 2014)*. IEEE. 2014, pp. 232–238.

[19] Gabjo Kim et al. “Technology clusters exploration for patent portfolio through patent abstract analysis”. In: *Sustainability* 8.12 (2016), p. 1252.

[20] A. Kumar and M. Jain. *Ensemble Learning for AI Developers: Learn Bagging, Stacking, and Boosting Methods with Use Cases*. Apress, 2020. ISBN: 9781484259399.

[21] Jieh-Sheng Lee and Jieh Hsiang. “PatentBERT: Patent Classification with Fine-Tuning a pre-trained BERT Model”. In: *arXiv e-prints* (2019), arXiv–1906.

[22] Loet Leydesdorff, Duncan Kushnir, and Ismael Rafols. “Interactive overlay maps for US patent (USPTO) data based on International Patent Classification (IPC)”. In: *Scientometrics* 98.3 (2014), pp. 1583–1599.

[23] Shaobo Li et al. “DeepPatent: patent classification with convolutional neural networks and word embedding”. In: *Scientometrics* 117.2 (2018), pp. 721–744.

[24] Aristidis Likas, Nikos Vlassis, and Jakob J Verbeek. “The global k-means clustering algorithm”. In: *Pattern recognition* 36.2 (2003), pp. 451–461.

[25] Yinhan Liu et al. “Roberta: A robustly optimized bert pretraining approach”. In: *arXiv preprint arXiv:1907.11692* (2019).

[26] Google LLC. *Colaboratory Frequently Asked Questions*. 2022. URL: https://research.google.com/colaboratory/faq.html (visited on 04/12/2022).

[27] Scott Menard. *Applied logistic regression analysis*. Vol. 106. Sage, 2002.

[28] Yu Meng et al. “Text Classification Using Label Names Only: A Language Model Self-Training Approach”. In: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. 2020, pp. 9006–9017.

[29] George A Miller. “WordNet: a lexical database for English”. In: *Communications of the ACM* 38.11 (1995), pp. 39–41.

[30] G. Nandi and R.K. Sharma. *Data Science Fundamentals and Practical Approaches: Understand Why Data Science Is the Next*. BPB Publications, 2020. ISBN: 9789389845662.

[31] Alireza Noruzi and Mohammadhiwa Abdekhoda. "Google Patents: The global patent search engine". In: *Webology* 11.1 (2014).

[32] United States Patent and Trademark Office. *CPC Definition - Subclass H01M*. 2022. URL: https://www.uspto.gov/web/patents/classification/cpc/html/defH01M.html (visited on 05/09/2022).

[33] Srebrovic Rob and Yonamine Jay. "Leveraging the BERT algorithm for Patents with TensorFlow and BigQuery". In: (Oct. 2020). URL: https://services.google.com/fh/files/blogs/bert_for_patents_white_paper.pdf.

[34] Ingo Ruczinski, Charles Kooperberg, and Michael LeBlanc. "Logic regression". In: *Journal of Computational and graphical Statistics* 12.3 (2003), pp. 475–511.

[35] Scikit-learn. *sklearn.linear$_m$odel.LogisticRegression*. 2022. URL: https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html (visited on 04/25/2022).

[36] Eva Sharma, Chen Li, and Lu Wang. "BIGPATENT: A Large-Scale Dataset for Abstractive and Coherent Summarization". In: *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. 2019, pp. 2204–2213.

[37] Svetlana Sheremetyeva. "Natural language analysis of patent claims". In: *Proceedings of the ACL-2003 workshop on Patent corpus processing*. 2003, pp. 66–73.

[38] Steven S Skiena. *The data science design manual*. Springer, 2017.

[39] Rob Srebrovic. *BERT for Patents*. 2020. URL: https://github.com/google/patents-public-data/blob/master/models/BERT%5C%20for%5C%20Patents.md (visited on 04/18/2022).

[40] Zhiqing Sun et al. "MobileBERT: a Compact Task-Agnostic BERT for Resource-Limited Devices". In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. 2020, pp. 2158–2170.

[41] Kanae Takahashi et al. "Confidence interval for micro-averaged F1 and macro-averaged F1 scores". In: *Applied Intelligence* 52.5 (2022), pp. 4961–4972.

[42] W.H. Tok, A. Bahree, and S. Filipi. *Practical Weak Supervision*. O'Reilly Media, 2021. ISBN: 9781492077015. URL: https://books.google.se/books?id=mLpFEAAAQBAJ.

[43] Anthony Trippe. "Guidelines for preparing patent landscape reports". In: *Patent landscape reports. Geneva: WIPO* (2015), p. 2015.

[44] Bruno Van Pottelsberghe de la Potterie and Malwina Mejer. "The London Agreement and the cost of patenting in Europe". In: *European Journal of Law and Economics* 29.2 (2010), pp. 211–237.

[45] Ashish Vaswani et al. "Attention is all you need". In: *Advances in neural information processing systems* 30 (2017).

[46] Suzan Verberne, Cornelis H. A Koster, and Nelleke Oostdijk. "Quantifying the Challenges in Parsing Patent Claims". In: *In Proceedings of the 1st International Workshop on Advances in Patent Information Retrieval*. 2010, pp. 14–21.

[47] Zihan Wang, Dheeraj Mekala, and Jingbo Shang. “X-Class: Text Classification with Extremely Weak Supervision”. In: *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. 2021, pp. 3043–3053.

[48] Adrienne Yapo and Joseph Weiss. “Ethical Implications of Bias in Machine Learning”. In: *Proceedings of the 51st Hawaii International Conference on System Sciences*. 2018.

[49] Guoqiang Peter Zhang. “Neural networks for classification: a survey”. In: *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)* 30.4 (2000), pp. 451–462.

[50] Hongyi Zhang et al. “mixup: Beyond Empirical Risk Minimization”. In: *International Conference on Learning Representations*. 2018.

[51] Lu Zhang et al. “Weakly-supervised Text Classification Based on Keyword Graph”. In: *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*. 2021, pp. 2803–2813.

[52] Xiaojin Zhu. “Semi-Supervised Learning Literature Survey”. In: *world* 10 (2005), p. 10.

A

Appendix 1: LOTClass Keywords

A.1 Manufacturing and Applications

Keywords

Manufacturing:
Manufacture, manufacturing, production, create, synthesis, masking, functionalization, characterization.

Application:
Application, apply, enhanced, energy, electronics, biotech, enviromental, capacitors, batteries, cells, power, functional, composites.

Category Vocabulary

Manufacturing:
Manufacture, production, preparation, manufacturing, processing, producing, generation, synthesis, reaction, development, produce, manufactured, conversion, reduction, treatment, control, produced, formation, purification, forming, making, work, productions, driving, mass, technology, recovery, productivity, fabrication, yield, fixing, generating, growth, mounting, obtaining, preparing, heating, performing, handling, cooling, assembling, manufactures, wiring, plating, developing, peeling, using, cleaning, selection, lighting, compression, creation, molding, scanning, drying, maintenance, deposition, technical, producer, improving, selecting, method, expression, requirement, modification, extraction, scale, detection, rotation, separation, operating, peripheral, manufacturer, covering, switching, quality, reproduction, sealing, discovery, polymerization, building, winding, feeding, synthetic.

Application:
Application, energy, applications, power, applied, apply, electricity, applicability, applying, information, analysis, injection, item, current, absorption, object, resource, energies, applicant, input, activity, patent, ion, addition, ability, document, output, assignment, submission, electronic, electron, file, implementation, number, energetic, battery, app, electronics, powers, content, grant, batteries, cells, end, disclosure, voltage, force, specification, order, alternative, oil, request, efficiency, storage, utility, reference, area, electrical, electric, article, action, service, emission, utilization, materials, connection, pressure, fuel, entry, environment, extension, source, devices, light, heat, adaptation, sensors, usage, charge, identification.

A.2 Batteries, Fuel Cells, and Capacitors:

Keywords

Batteries:
Batteries.

Fuel cells and capacitors:
Capacitor, capacitors, fuel, cell, cells.

Category Vocabulary

Batteries:
Batteries, tires, databases, antennas, alloys, roofs, bolts, cases, reservoirs, plants, computers, relays, medicines, poles, switches, compounds, cushions, buttons, salts, cities, boxes, powers, drugs, parents, screens, shells, lids, algorithms, packs, blocks, services, paints, transformers, cosmetics, memories, things, cabinets, bases, states, meters, watches, buses, plates, arrangements, polymers, ceramics, methods, derivatives, preparations, ions, ones, alternatives, filters, solutions, rubbers, lamps, networks, gears, warehouses, belts, phones, motors, mats, stocks, displays, complexes, limits, models, fences, engines, sources, situations, combinations, stores, shocks, reports, nets, metals, currents, plans, pumps, variables, protections, bearings, rails, schedules.

Fuel cells and capacitors:
Cell, cells, device, unit, sensor, combination, capacitor, film, fuel, capacitance, memory, material, method, electrode, condenser, capacitive, carbon, power, capacity, membrane, polymer, composite, storage, compound, flow, diode, structure, dielectric, layer, precursor, cycle, direct, product, contact, panel, pump, end, gas, condition, chemical, filter, container, vehicle, core, electrolyte, voltage, cup, cleaner, path, element, single, reaction, fluid, super, space, supply, cellular, system, ratio, ceramic, hydrogen, refrigerator, fabric, stack, blood, products, heat, methanol, composition, process, store, loop, flexible, units, barrier, tube, component, separator, fiber, air, water, conditions, line, enzyme, catalyst, panels.

DEPARTMENT OF MATHEMATICAL SCIENCES
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden
www.chalmers.se

CHALMERS
UNIVERSITY OF TECHNOLOGY