Hierarchical classification of South African research outputs

Jaco du Toit
Stellenbosch University
Stellenbosch, Western Cape, South Africa
22808892@sun.ac.za

Marcel Dunaiski
Stellenbosch University
Stellenbosch, Western Cape, South Africa
marceldunaiski@sun.ac.za

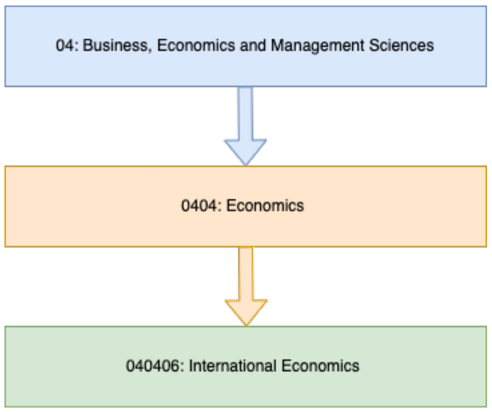
Abstract

The Department of Higher Education and Training (DHET) maintains a hierarchical structure of categories that represent the various fields of study in South African higher education. Although some South African universities classify their research outputs according to the categories specified in this document, many do not do so consistently. The objective of this project is to develop a hierarchical classifier for the categorisation of South African master's theses and PhD dissertations according to the official documentation provided by the DHET by using only the title and abstract of the associated document. Categorisation of these documents is a difficult problem due to the large number of possible categories and the semantic nature of abstracts which often contain information related to several different fields of study. We use two models to extract semantic feature representations from a text document: a transformer-based language model and a Latent Dirichlet Allocation (LDA) topic model. We combine the information obtained from these models through different fusion mechanisms and use the result to train a number of classification models which form the hierarchical classifier. We evaluate several Convolutional Neural Network (CNN) architectures which includes a standard CNN, multi-channel CNN and two types of attention mechanisms as extensions to the standard CNN architecture. For comparison, we also evaluate a Long Short-Term Memory (LSTM) Recurrent Neural Network (RNN) classification model. We perform experiments with the different feature extraction strategies and classification models to determine which combination performs best for the task at hand. We find that the multi-channel CNN outperforms the other classification models and is thus the most suitable classification model for the associated problem with the given feature extraction methods. Furthermore, we find that the combination of local semantic and global topic information provided by the transformer-based and LDA models respectively, obtains the best performance among the considered feature extraction methods.

Keywords: Hierarchical text classification, LDA, BERT, CNN

1 Introduction

The Classification of Educational Subject Matter (CESM) is a document which categorises the different fields of study in South African higher education in a hierarchical structure and is maintained by the Department of Higher Education and Training (DHET). Figure 1 depicts an example of the hierarchical structure of the CESM specification and the codes used for the different categories regarding International Economics. Here, the most broad category is assigned the base code and the code is extended as more detail is added to the lower levels of the hierarchy. Many South African universities do not consistently provide categorisation of their research outputs according to the CESM standard on their publicly accessible institutional repositories. This project aims to develop an accurate and reliable method to assign potentially multiple labels in a hierarchical structure to South African master's theses and PhD dissertations according to the official CESM documentation. It is important for universities and researchers to maintain consistent labels of these documents in order to categorise them in a structured manner. Taking the hierarchical structure of the CESM specification into account further allows us to classify these documents with varying levels of detail depending on the associated subject. The hierarchical classifier developed in this project can be used as a category recommender system which suggests the most suitable categories for an associated document from which a human expert can determine the final category.


04: Business, Economics and Management Sciences
0404: Economics
040406: International Economics
Figure 1. CESM hierarchy for International Economics.

Text classification (TC) is an important and challenging task which is used in many different fields where text documents need to be categorised. Feature extraction is one of the most important components of text classification and it is the process of extracting a semantic representation of a text document which can be used to train classification models. With the introduction of transformer-based language models [44], new methods have been developed to solve these problems which obtain better results than traditional methods used for text classification [28]. Peinelt *et al.* [34] introduced an approach which combines information obtained from transformer-based language models and topic extraction models [4] to perform semantic similarity detection between documents. This technique has shown good results for TC tasks in a recent paper by Liu *et al.* [28] where they used the ALBERT [22] language model and the Latent Dirichlet Allocation (LDA) [5] topic extraction model for multi-label text classification problems.

Hierarchical text classification (HTC) extends TC problems by assigning categories to text documents in a hierarchical tree structure. Many techniques have been developed to solve HTC problems, with the most simple approach being the conversion to a flat multi-label classification problem which does not consider the hierarchical structure of the labels and only classifies leaf nodes. However, more advanced approaches have been developed to leverage the hierarchical structure of the classes which have shown improvements for HTC problems [41]. Stein *et al.* [41] observed that even though significant advancements have been made in TC problems through the use of word embeddings as feature representations of documents, these advancements had not been translated to HTC problems. They performed experiments with different word embedding generation methods such as GloVe, word2vec and fastText and found that the use of word embeddings is a very promising approach for HTC. Furthermore, there have been many attempts at solving HTC problems in the domain of categorising medical text notes in a hierarchical structure to assist with the diagnosis of illnesses [24, 31, 42, 43, 45]. In these papers several machine learning techniques such as Convolutional Neural Networks (CNNs), Recurrent Neural Networks (RNNs), transformer models and attention mechanisms were used to classify medical text notes. Strydom [42] used many of these techniques with a focus on the use of attention mechanisms in CNNs, RNNs and transformer models and obtained very good results on the associated task. Strydom obtained an F1 score of 0.4989 on the MIMIC II dataset and outperformed the state-of-the-art JointLAAT model [45] by 0.0079. Furthermore, Strydom outperformed the LAAT model proposed by Vu *et al.* [45] on the MIMIC III dataset and set a new benchmark F1 score of 0.5856.

Our approach to the categorisation of South African research outputs is to develop a hierarchical classifier which utilises the feature extraction methods used by Liu *et al.* [28] and several of the classification model architectures used by Strydom [42]. The classifier uses the title and abstract of a document to obtain a semantic representation of the associated document as a whole. We use two models to obtain semantic feature representations: a transformer-based language model and a LDA topic model to extract local semantic and global topic information respectively. We evaluate three fusion mechanisms which combine the information obtained from these models in different ways. We use the resulting feature representations to train different neural network architectures which form the implementation of the hierarchical classifier. Specifically, we evaluate several CNN and RNN architectures to extract different patterns and features from the text document such that it can be hierarchically categorised. Furthermore, we evaluate two types of attention mechanisms which are added as extensions to the standard CNN architecture.

The rest of this paper proceeds as follows: Section 2 provides the required background information on the main concepts related to this paper. Section 3 discusses related research on multi-label and hierarchical text classification. Section 4 describes our approach to developing the hierarchical classifier and our evaluation methodology. Lastly, in Section 5, we present and discuss the results obtained by the evaluated models.

2 Background

This section presents background information on the main concepts used to develop our hierarchical classifier. We describe the different types of label classification problems, the two models used to obtain semantic feature representations of the text documents and the different classification model architectures that are used in this project.

2.1 Label classification problems

The traditional single-label classification problem aims to learn from a set of instances where each instance is associated with a unique class label from a set of disjoint class labels $\mathcal{L}$ [40]. The single-label classification problem is classified as a binary classification problem (when $|\mathcal{L}| = 2$) or a multi-class classification problem (when $|\mathcal{L}| > 2$). Multi-label classification (MLC) allows each instance to be associated with more than one class from the set of labels $\mathcal{L}$. The goal of MLC is to learn from a set of instances where each instance belongs to potentially multiple classes in $\mathcal{L}$. Multi-label text classification (MLTC) is an instance of MLC and Liu *et al.* [28] defines it as the process of allocating a set of labels that are most relevant to a document from an overall label set to reflect the semantic information of the associated document. Hierarchical text classification is an extension of this concept where potentially multiple labels are assigned to a document in a hierarchical structure where each level of the hierarchy provides additional detail.

2.2 Topic models

Topic models [4] are unsupervised machine learning models that extract abstract topics from a corpus of documents.

Topic models consist of probabilistic latent semantic analysis (PLSA) [17], LDA and extensions of these models. LDA is one of the most popular topic models currently in use [28] and it is the topic model used in this project.

LDA [5] is a probabilistic generative topic model used to find topics related to a document. LDA is based on the Bag of Words (BOW) model [48] which represents a text document as a multiset of its words and ignores order and context information. The LDA model represents each topic as a probabilistic distribution over a set of words contained in the corpus of documents. The documents are represented as a probabilistic distribution over a set of topics and thus the most relevant topics associated with each document can be determined based on the assumption that documents with similar topics will use a similar group of words.

The LDA algorithm is provided with a fixed number of K topics to discover in a corpus of documents and has the objective of learning the topic distribution of each document and the word distribution of each topic. The algorithm goes through every document and randomly assigns each word in a document to a topic. These assignments of words to topics are used to form the topic and document representations as the topic distribution is the distribution of the words assigned to that topic and the document distribution is the distribution of the topics assigned to the words of the document. The algorithm attempts to reassign topics to words such that the abstract topic representations become representative of the topics found in the corpus of documents. This is done through the following procedure:

1. For every document calculate the proportion of words in document D that are currently assigned to topic T as $p(\text{Topic } T \mid \text{Document } D)$ for $T \in \{1, 2, \dots, K\}$.
2. For every word in the corpus of documents calculate the proportion of assignments of the word W to the topic T as $p(\text{Word } W \mid \text{Topic } T)$ for $T \in \{1, 2, \dots, K\}$.
3. For each word W in document D compute the probability $p(\text{Topic } T \mid \text{Document } D) \cdot p(\text{Word } W \mid \text{Topic } T)$ for $T \in \{1, 2, \dots, K\}$ and assign the topic T with the highest probability to that word.
4. Repeat the steps above for a large number of iterations until an approximately steady state of assignments that are acceptable is reached.

2.3 Transformer-based language models

Transformer models [44] are deep learning models that use a mechanism called self-attention to differentially weigh the significance of each part of the input data. These attention mechanisms allow modelling of dependencies without regarding the distance between them in the input or output sequence. The transformer model architecture consists of components called encoders and decoders [44]. Bidirectional Encoder Representations from Transformers (BERT) [12] is a transformer-based language model that consists of a sequence of encoders and has shown to improve on previous state-of-the-art solutions on various natural language processing (NLP) tasks, including text classification.

The BERT model uses a deep bidirectional model as opposed to the traditional left-to-right model or the concatenation of a left-to-right and right-to-left model used for pre-training language models. The model is pre-trained through two unsupervised tasks, mask language modelling (MLM) and next sentence prediction (NSP). BERT trains deep bidirectional representations through the MLM model by masking a percentage of the input tokens randomly and predicting these masked tokens. NSP is used to pre-train the model by predicting the relationship between two sentences which is important for downstream tasks such as Question Answering (QA) and Natural Language Inference (NLI). BERT uses these techniques to extract local semantic information by using the surrounding text of a given word to establish context. The BERT model uses a special token known as the CLS token for which an embedding is learned such that the token can be used as a vector representation of the input sequence as a whole for downstream tasks that require such representations.

2.4 Artificial neural networks

Artificial neural networks (ANNs) are machine learning algorithms inspired by the working of neurons in the human brain. An ANN consists of units called artificial neurons (ANs) which implement a functional mapping from a number of input signals to an output signal. An AN receives input signals from other neurons in the network with a certain weight associated with each input signal. The output of the AN is most commonly calculated through passing the weighted sum of the inputs combined with a bias unit through a function known as the activation function. Suppose an AN with an activation function f receives input signals in the form of a vector $\mathbf{x} = [x_1, x_2, \dots, x_n]$ with associated weights $\mathbf{w} = [w_1, w_2, \dots, w_n]$ and bias unit b. The output of the AN is given by

$$f((\sum_{i=1}^{n} w_i x_i) + b) = f(\mathbf{w}^T \mathbf{x} + b) \tag{1}$$

The associated weights ($\mathbf{w}$) and bias unit (b) are learned during training through a procedure known as backpropagation [26]. Backpropagation is an algorithm that propagates the error of output nodes back through the layers of the ANN to adjust its weights and biases in order to minimise an error function which compares the output of the ANN to the expected outputs. Thus, backpropagation is the mechanism that the ANN uses to learn suitable weights and biases to fit the model to the training data. The activation function determines the effect of an AN and non-linear activation functions allow ANNs to learn more complex functional mappings from the input space to the output space through the introduction of non-linearity. The rectified linear unit (ReLU) function has become the most widely used activation function in deep learning networks [32] and is given by

$$\mathrm{ReLU}(x) = \max(0, x) \tag{2}$$

Feed-forward neural networks (FFNNs) form the basis architecture for most neural networks and are structured as described above where connections between ANs occur only in one direction from the input neurons to the next layer in the ANN and weights and biases are learned through backpropagation. Other popular ANN architectures include CNNs and RNNs.

2.5 Convolutional neural networks

A CNN [23] is a specialised deep learning model which uses a convolutional operation to extract different patterns and features of an input feature. The standard CNN consists of a Convolution Layer (CL), which has the objective of extracting high-level features and patterns, a Pooling Layer (PL) which reduces the dimension size of the data, and a Fully Connected Layer (FCL) which forms the output of the model.

CNNs have become popular machine learning methods for text classification after a paper by Kim [18] improved the previous state-of-the-art in various TC problems. The CL layer extracts patterns and features from an input text sequence, where each word has a vector embedding representation, by sliding a filter across the input to group certain words together. The feature value for each of the groups is calculated as the weighted sum of the word embeddings under the filter with the weights of the filter itself. The CNN randomly initialises the values of the filter matrix and learns them during training. The CNN uses a certain number of these filters such that each filter learns different patterns from the input text sequence to produce a number of feature vectors. The PL uses a summary statistic of nearby outputs to decrease the dimension of the representation and the required computational effort. Max-pooling is the most popular pooling function which uses the maximum output of a neighbourhood. The output of the PL is fed into the FCL which is a standard feedforward layer. The CNN learns the values of the different filters and the weights and biases of the FCL through a backpropagation mechanism to minimise an error function based on the output of the FCL.

2.6 Recurrent neural networks

RNNs [38] are a class of ANNs which are able to model sequence data by allowing connections between nodes to form a cycle. Suppose we have a vector representation of a document as $\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_T]$ where T is the number of words in the document. At a certain timestep t, the model considers the input $\mathbf{x}_t$ as well as the hidden state of the previous layer. The hidden state of the previous layer ($\mathbf{h}_{t-1}$) holds information from all the preceding timesteps. The hidden state of the current timestep ($\mathbf{h}_t$) is calculated as

$$\mathbf{h}_t = f(\mathbf{W}^{(h)}\mathbf{h}_{t-1} + \mathbf{W}^{(x)}\mathbf{x}_t) \tag{3}$$

where f is the chosen activation function and $\mathbf{W}^{(h)}$ and $\mathbf{W}^{(x)}$ are weight matrices associated with the previous hidden state and the current input respectively. Thus, a document can be represented by the final hidden state ($\mathbf{h}_T$) which provides a representation of the document as a whole. The standard RNN architecture is prone to suffer from the vanishing gradient problem which is a result of the backpropagation process that continuously multiplies derivatives back through the network such that the gradients become very small. This implies that the network can not effectively consider the outputs of the earlier timesteps when sequences grow long.

A long short-term memory network (LSTM) [16] is a type of RNN that was introduced to combat the vanishing gradient problem such that longer sequences can be effectively modelled. LSTMs maintain a memory state which is referred to as a cell state. The cell state is used to store long-term information that it deems as important and this information is maintained by three gate vectors: an input ($\mathbf{i}_t$), output ($\mathbf{o}_t$) and forget ($\mathbf{f}_t$) gate. The input gate determines how much incoming information is used to update the cell state and the forget gate determines which information is retained from the previous cell state. The output gate determines which information from the cell state is used in the hidden state. These gates use a sigmoid function which is given by

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{4}$$

such that the gate vectors are calculated as

$$\begin{aligned} \mathbf{i}_t &= \sigma(\mathbf{W}^{(i)}\mathbf{h}_{t-1} + \mathbf{U}^{(i)}\mathbf{x}_t + \mathbf{b}^{(i)}) \\ \mathbf{f}_t &= \sigma(\mathbf{W}^{(f)}\mathbf{h}_{t-1} + \mathbf{U}^{(f)}\mathbf{x}_t + \mathbf{b}^{(f)}) \\ \mathbf{o}_t &= \sigma(\mathbf{W}^{(o)}\mathbf{h}_{t-1} + \mathbf{U}^{(o)}\mathbf{x}_t + \mathbf{b}^{(o)}) \end{aligned} \tag{5}$$

where the $\mathbf{W}$ and $\mathbf{U}$ matrices are weight matrices and $\mathbf{b}$ is a bias vector associated with each of the different gates. The new cell information is calculated as

$$\tilde{\mathbf{c}}_t = \tanh(\mathbf{W}^{(c)}\mathbf{h}_{t-1} + \mathbf{U}^{(c)}\mathbf{x}_t + \mathbf{b}^{(c)}) \tag{6}$$

where tanh is a function given by

$$\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \tag{7}$$

The cell state is calculated as

$$\mathbf{c}_t = \mathbf{f}_t \otimes \mathbf{c}_{t-1} + \mathbf{i}_t \otimes \tilde{\mathbf{c}}_t \tag{8}$$

where $\otimes$ is the element-wise multiplication operator. Finally, the output gate is calculated as

$$\mathbf{h}_t = \mathbf{o}_t \otimes \tanh(\mathbf{c}_t) \tag{9}$$

2.7 Attention mechanisms

The attention mechanism was first developed by Bahdanau *et al.* [2] for the task of neural machine translation (NML) using RNN architectures. The intuition for the attention mechanism in NML problems is based on the way humans process text, by placing attention on specific words in the text input while considering the overall context of the input [42]. Even though the attention mechanism was originally developed for RNN architectures, a general definition of these mechanisms is provided by Vaswani *et al.* [44]. In the generalised definition of an attention mechanism we consider a query **Q** and a set of key-value pairs **K** and **V**. Attention weights ($\boldsymbol{\alpha}$) are computed between the query **Q** and each of the keys in **K** by calculating an alignment score between these vectors and passing the result through a softmax function,

$$\boldsymbol{\alpha} = \text{softmax}(a(\mathbf{Q}, \mathbf{K})) \tag{10}$$

where a is referred to as an alignment or compatibility function and the softmax function for the i-th element of an input **t** is defined as

$$\text{softmax}(\mathbf{t})_i = \frac{e^{t_i}}{\sum_{j=1}^{D} e^{t_j}} \tag{11}$$

where D is the number of elements in **t**. The alignment function aims to measure the similarity between the given query and key to determine which parts of the vector **V** related to that key to place attention on. The two alignment functions used in this project are referred to as the dot product alignment function given by

$$a(\mathbf{Q}, \mathbf{K}) = \mathbf{QK} \tag{12}$$

and the general alignment function given by

$$a(\mathbf{Q}, \mathbf{K}) = \mathbf{QWK} \tag{13}$$

where **W** is learned during training.

The output can then be calculated as a weighted sum over the values in **V** and $\boldsymbol{\alpha}$ given by

$$\mathbf{c} = \boldsymbol{\alpha}\mathbf{V} \tag{14}$$

The intuition behind this process is that each element of $\boldsymbol{\alpha}$ represents how much attention is placed on each element of **V**.

3 Related research

This section reviews the related research regarding MLC and hierarchical classification. The techniques for solving MLC problems include traditional machine learning methods such as problem transformation and algorithm adaption and more recently deep learning models such as CNNs, RNNs, as well as transformer-based models. Furthermore, hierarchical classification is an extension of MLC and many different approaches to solve such problems have been proposed.

3.1 Traditional machine learning methods

Problem transformation methods attempt to solve MLC problems by converting them to single-label classification problems. Binary relevance (BR) [7] is a problem transformation method which partitions MLC problems into several binary classification problems. The results from the independent binary classification problems are combined to provide a solution for the MLC problem. However, BR may lead to poor performance due to its assumption that labels are independent [40]. According to Bogatinovski *et al.* [6] BR does not perform well for problems with a large number of labels and the training of the method may be inefficient as it builds a model for each label. Label Powerset (LP) [1] is another problem transformation method that transforms MLC problems into multi-class classification problems such that each unique label-set is treated as a separate class. A multi-class classifier is then applied to solve the MLC problems. Although LP preserves the relationships between labels by grouping them in the same class, this method is prone to underfitting as the number of unique label-sets grows large [6].

Algorithm adaptation methods attempt to solve MLC problems by adapting existing learning techniques and applying them to the associated MLC data. Multi-label k Nearest Neighbour (MLkNN) [49] is an algorithm adaption method that adapts the Nearest Neighbour [14] method and applies it to MLC problems. This method measures the occurrences of all classes among the k nearest neighbours of a target instance to evaluate how likely the presence of a label is for the associated instance [1]. However, the performance of MLkNN is limited due to the fact that label correlations are not taken into account [1]. Multi-label decision tree (ML-DT) [1] is another algorithm adaptation method which adapts decision tree algorithms and applies it to MLC problems. The decision tree is built by using a multi-label entropy information gain criterion [28]. However, the calculation of the multi-label entropy in these methods assumes that labels are independent which may lead to poor performance [28].

3.2 Deep learning methods

Kim *et al.* [18] introduced a CNN model called TextCNN which uses a CNN model for text classification and sentence-level classification. According to Liu *et al.* [28] the TextCNN model has the disadvantage of fixed windows in CNNs and thus struggles to model long sequence information.

Chen *et al.* [9] developed an approach which uses a CNN to obtain a text feature sequence and feeds the result into a RNN to obtain a predicted label for the corresponding document. However, according to Liu *et al.* [28] this model is sensitive to the size of the training set and small training sets could lead to overfitting. Furthermore, most RNN-based solutions to MLTC problems implement the sequence-to-sequence framework which considers the relationships between the labels using sequence generation. This often leads to a greater impact of incorrect labels as former labels are often dependent on later labels [28].

The use of transformer-based models for MLTC problems has become popular with the inception of language models such as BERT and GPT [35]. Yarullin *et al.* [47] proposed a sequence-generating BERT model which uses BERT as an encoder for the sequence-to-sequence framework. They showed that the sequence-generating BERT model obtained similar accuracy with much less training time compared to the BERT model alone. Liu *et al.* [28] states that transformer-based models usually outperform CNN and RNN models for MLTC problems but due to the large number of parameters required and the complex network structure, there exist some limitations in practical applications.

Liu *et al.* [28] proposed a promising solution to solve MLTC problems inspired by the work of Peinelt *et al.* [34] which uses a combination of pre-trained contextual representations from BERT and information from a topic model to obtain a semantic representation of a text document. Liu *et al.* [28] developed the tALBERT-CNN method which combines the ALBERT [22] and LDA models to obtain local and global semantic information of documents respectively. This model outperformed other popular methods such as TextCNN [18] and XML-CNN [27] on three different datasets. They found that this fusion of information from a topic model and transformer-based language model improves text classification performance.

3.3 Hierarchical classification

Several strategies have been developed for the task of hierarchical classification which can be broadly categorised into three approaches: flat, local and global classification. Each of these strategies can be used to solve hierarchical text classification (HTC) problems where text documents are classified in a categorical hierarchy.

The flat classification approach is the simplest strategy to solve hierarchical classification problems as it does not consider the class hierarchy and only predicts leaf node classes. This approach is effectively the same as a standard classification method but it solves the hierarchical classification problem by assuming that when a leaf node is assigned to a class, all of the higher-level classes are implicitly assigned to that class [41]. Fall *et al.* [15] used a flat classification approach along with decision trees and Naive Bayes to classify text documents in a categorical hierarchical structure. This approach is not able to handle problems with non-mandatory leaf node predictions and has the disadvantage of building a classifier which has to differentiate between a large number of leaf classes without using information about parent-child relationships that can be found in the hierarchical structure of the data [41].

The first local classifier approach was developed by Koller and Sahami [11] which has since been expanded to different methods of solving hierarchical classification problems. These methods take into account the hierarchical structure of the data and use the local information perspective to more effectively solve hierarchical problems. The most common local classifier approaches are the local classifier per node, per parent node, and per level approaches. These three approaches differ in the way they are trained, but perform a similar top-down testing phase where the algorithm first predicts the top-level and uses the predicted class to narrow the choices of the classes to predict in the deeper levels. Thus, these approaches have the disadvantage that an incorrectly predicted class at a certain level in the hierarchy will lead to incorrect predictions in all of the lower levels of the hierarchy. The local classifier per node approach trains a binary classifier for each node in the hierarchical class structure, apart from the root node, where the classifier of a node predicts whether an instance belongs to that class or not. The local classifier per parent node approach trains a multi-class classifier for each parent node in the hierarchical class structure, where the parent node classifier is used to distinguish between the classes of its child nodes. The local classifier per level approach trains a multi-class classifier for each level of the class hierarchy and each classifier predicts one or more classes in its associated level. Koller and Sahami [11] introduced a model which considered the hierarchical structure of categories and consisted of a set of Bayesian classifiers, with one for each hierarchy node. This approach showed a significant improvement on previous models [41]. Cesa-Bianci *et al.* [8] introduced a hierarchical support vector machine (SVM) approach where SVM learning was applied to categories if their parent categories were predicted. More recently, neural network-based local approaches have been used to solve HTC problems. Ruiz and Srinivasan [30] introduced a method which uses a set of NN models and a divide and conquer approach to obtain smaller classification problems based on the hierarchical structure. They found that their solution improved on flat text classification approaches by considering the hierarchical structure. Stein *et al.* [41] performed several experiments to compare the performance of different word embedding generation methods such as GloVe, word2vec and fastText to obtain the semantic representation of a document for HTC problems. Furthermore, they used different classifier architectures and compared the performance of a local classifier per node approach to a flat classification approach. They found that the use of word embeddings is a very promising approach to HTC problems and that the local classifier approach outperformed the flat classification approach.

The global classifier approach builds a single classification model which considers the overall hierarchical class structure. Silla and Freitas [41] state that although there is no specific main characteristic that is shared by all global classifier approaches, they all take into account the whole class hierarchy at once and do not have the modular local training found in local classifier approaches. Kiritchecnk *et al.* [20] developed a global classifier approach which is based on converting hierarchical classification problems to multi-label classification problems which considers all of the classes in the hierarchical class structure by augmenting the non-leaf nodes with information related to its ancestor classes. This approach may lead to class-prediction inconsistency in the hierarchy and a post-processing step is required to ensure that the hierarchical structure is followed. Another global classifier approach is based on the Rocchio classifier [39] which uses clustering to assign new instances through some distance measure between the query instance and each class. This approach was implemented by Labrou and Finin [21] to classify web pages into the hierarchical categories provided by Yahoo! using different types of category descriptions.

One of the most popular applications of HTC is the categorisation of medical text notes to assist with the diagnosis of illnesses. Several approaches have been developed for this particular problem which have advanced the state-of-the-art for HTC tasks [24, 31, 42, 43, 45]. These approaches use various machine learning techniques such as CNNs, RNNs, transformer models and attention mechanisms. Strydom [42] used these techniques with a focus on using attention mechanisms in these machine learning algorithms for the categorisation of medical text notes. Strydom outperformed the LAAT model [45] on the MIMIC III dataset and set a new benchmark F1 score of 0.5856. Furthermore, Strydom obtained an F1 score of 0.4989 on the MIMIC II dataset and outperformed the state-of-the-art JointLAAT model proposed by Vu *et al.* [45] by 0.0079.

4 Methodology

This section discusses our approach to develop a hierarchical classifier for South African research outputs. We discuss the characteristics of the dataset, feature extraction methods, hierarchical classifier architecture, and the different classifier model architectures. Furthermore, we describe the evaluation metrics used to measure the performance of the models.

4.1 Dataset and preprocessing

The dataset contains the titles and abstracts of 9917 South African master's theses and PhD dissertations and the CESM categories assigned to each document. Table 1 provides a few examples of dataset entries and the components they consist of. Here, we see that some entries are labelled with all three levels of the hierarchy, while others are only labelled with the most broad category they belong to. We performed one-hot encoding on the CESM codes assigned to the documents such that these categorical variables can be used in the machine learning models. Thus, each document $i \in \{1, 2, \ldots, N\}$ where N is the number of documents, is associated with a vector $\mathbf{y}_i$ which represents the categories the document belongs to from the set of categories $k \in \{1, 2, \ldots, K\}$, where K is the number of classes, and is determined by

$$y_{i,k} = \begin{cases} 1, & \text{if document } i \text{ belongs to class } k \\ 0, & \text{otherwise} \end{cases} \tag{15}$$

We observed that the class distributions of the dataset are imbalanced. Thus, we removed classes that have less than a certain threshold of instances belonging to them and removed all instances that belong to these classes. We sampled equally from the remaining classes to prevent the model from becoming biassed towards classes that have a higher number of associated instances. Furthermore, we observed that the third level of the CESM hierarchy is too imbalanced and contains too little information to be included in the classification process. Thus, we only classify documents for the top two levels of the CESM hierarchy. The final set of classes included 11 first level categories and 39 second level categories, with each first level category having between two and five associated second level categories.

We used a 3-way split to train, tune and test our models. The models were trained and fitted with suitable parameter values using 70% of the dataset. We used 15% of the dataset as the validation set to provide an evaluation of the model fit along with the tuning of hyperparameters. Finally, 15% of the documents were reserved as a test dataset to provide an unbiased evaluation of the final model performance.

We preprocessed the corpus of documents which consist of the titles and abstracts of South African master's theses and PhD dissertations to remove all of the invalid entries that either do not have any valid label information or have abstracts that are not English. We further preprocessed the dataset separately for the two models used to extract semantic feature representations of the documents.

For the LDA topic model we executed the following procedure for each document: tokenization, removal of punctuation marks, conversion to lowercase, removal of stopwords and stemming. We used the preprocessed documents to form a dictionary containing mappings of ID's to words in the corpus. We used this dictionary along with the corpus to create a representation of each document by determining the frequency of each word in the document and mapping this frequency to the associated ID from the dictionary. The dictionary and the corpus frequency representation form the input to the LDA model.

The transformer-based language model tokenizes the input text, followed by a word embedding conversion which is ultimately used as input to the model. Furthermore, the model only accepts 512 words and thus only uses the first 512 words of each document which comprises the title followed by the first part of the abstract of the associated research

Table 1. Example dataset entries.

ID	Title	Abstract	CESM code 1	CESM code 2	CESM code 3
2301	"Understanding poverty..."	"This thesis..."	04	0404	040401
3465	"Microclimate and..."	"This study..."	01	-	-

output. Documents with fewer than 512 words are padded with special padding tokens.

For the label embedding attention mechanism we used the textual descriptions of the CESM categories as provided in the CESM document. We concatenated the third level descriptions to form the second level descriptions because the CESM document does not provide textual descriptions for the second level categories. These descriptions start with redundant phrases which were removed to improve the quality of the descriptions.

4.2 Feature extraction

We used two models to extract semantic feature vectors for each word in a document and the document as a whole and used different fusion mechanisms to combine the information obtained from these models. The two feature extraction models include the SciBERT language model and the LDA topic model.

We extracted local semantic information through the SciBERT model [3] which is a language model based on BERT that was pre-trained on scientific data, and is thus suitable for our application of classifying research outputs. The SciBERT model provides embeddings of words which take into consideration the context of that word in the sentence. These embeddings are fine grained and provide local context of the words in a document without considering the other documents in the corpus. The SciBERT model extracts the document-level information through a semantic vector, $\mathbf{Z}_i \in R^s$, where s is the size of the hidden state of the model. $\mathbf{Z}_i$ is obtained from the CLS token for each document $i \in \{1, 2, \ldots, N\}$, where N is the number of documents in the corpus. Furthermore, the SciBERT model provides a semantic vector, $\mathbf{w}_{ij} \in R^s$, for each word $j \in \{1, 2, \ldots, p\}$ in document i through the token associated with the word in the last layer of the model.

The LDA model provides global topic information of each word and document in the corpus. These embeddings consider the entire corpus and provide a global topic representation of each word and document by considering the context of the topics found in the corpus as a whole. The LDA model provides the document-level topic vector, $\mathbf{X}_i \in R^d$ where d is the number of topics, for each document $i \in \{1, 2, \ldots, N\}$. Furthermore, the LDA model obtains the topic word vector, $\mathbf{t}_{ij} \in R^d$, for each word $j \in \{1, 2, \ldots, p\}$ where p is the number of words in document i.

We used three different fusion strategies to combine the semantic vectors obtained from the two models where the result forms the input to the hierarchical classifier. The fusion strategies require the dimension of the vectors obtained from the SciBERT and LDA models to be the same. However, the hidden state of the SciBERT model has a size of 768 which is unreasonably high for the number of LDA topics and would result in less meaningful topics being formed. Thus, we chose d as the total number of categories (50) and padded the rest of the vector embedding with zeros.

We used the following fusion strategies:

- Strategy 1 (S1) uses the semantic word embeddings, $\mathbf{w}_{ij}$, obtained by the SciBERT model to represent the feature vector of document i. Thus, the feature vector for document i is given by

$$\mathbf{F}_i = \begin{bmatrix} \mathbf{w}_{i1} \\ \mathbf{w}_{i2} \\ \cdots \\ \mathbf{w}_{ip} \end{bmatrix} \in R^{p \times s} \tag{16}$$

- Strategy 2 (S2) concatenates the local semantic word embeddings, $\mathbf{w}_{ij}$, obtained by the SciBERT model and the global topic word embeddings, $\mathbf{t}_{ij}$, obtained by the LDA model. Thus, the feature vector for document i is given by

$$\mathbf{F}_i = \begin{bmatrix} \mathbf{w}_{i1} \\ \mathbf{w}_{i2} \\ \cdots \\ \mathbf{w}_{ip} \\ \mathbf{t}_{i1} \\ \mathbf{t}_{i2} \\ \cdots \\ \mathbf{t}_{ip} \end{bmatrix} \in R^{2p \times s} \tag{17}$$

- Strategy 3 (S3) concatenates the local semantic word and document embeddings, $\mathbf{w}_{ij}$ and $\mathbf{Z}_i$, obtained by the SciBERT model, as well as the global topic word and document embeddings, $\mathbf{t}_{ij}$ and $\mathbf{X}_i$, obtained by the LDA model. Thus, the feature vector for document i is given by

$$\mathbf{F}_i = \begin{bmatrix} \mathbf{w}_{i1} \\ \mathbf{w}_{i2} \\ \cdots \\ \mathbf{w}_{ip} \\ \mathbf{Z}_i \\ \mathbf{X}_i \\ \mathbf{t}_{i1} \\ \mathbf{t}_{i2} \\ \cdots \\ \mathbf{t}_{ip} \end{bmatrix} \in R^{2(p+1) \times s} \tag{18}$$

To provide a baseline for these three feature extraction strategies, we implemented a feature extraction method which learns an embedding for each word in the corpus during the training process. First, we encoded each unique word in the corpus to an integer and mapped them to a word embedding such that the same words have the same embedding. These word embeddings do not consider any context and are randomly initialised and learned along with the associated weights and biases during the training process of the model. We refer to this feature extraction method as S0.

4.3 Hierarchical classifier architecture

In order to classify the documents in a hierarchical structure, we used the local classifier per parent node approach which constructs a multi-class classifier for each non-leaf node in the hierarchical tree of categories. Figure 2 presents the structure of the hierarchical classifier, where the dashed rectangles indicate the constructed classifiers. As we only consider the top two levels of the CESM hierarchy, we construct a root classifier as well as a classifier for each category in the first level of the hierarchy. The root classifier is trained on all of the data and is used to predict the first level category of a document. For each classifier associated with a first level category, the classifier is trained on data labelled with the associated second level categories with the aim of predicting the instance as belonging to one of the second level categories. Thus, these models are only trained and used to distinguish between a small number of categories that fall under the first level category.

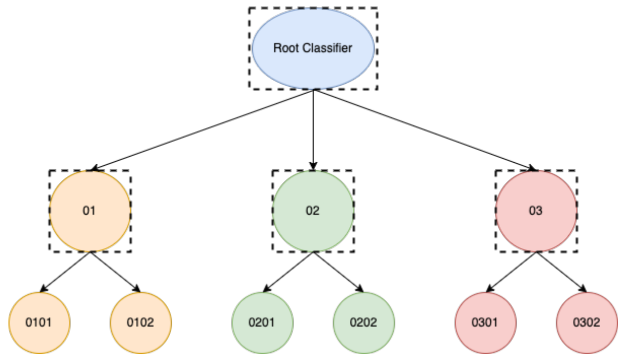


Figure 2. Hierarchical classifier.

When the labels of a test instance are predicted the following procedure is followed: the root classifier predicts the first level category of the instance by assigning the category with the highest level of certainty in the output of the model. This predicted category is then chosen as the first level category and the classifier associated with this category is used to predict the second level category of the test instance. As not all documents are assigned a category for the second level of the hierarchy, the category is only assigned if the level of certaintanty in the output of the model is above a certain threshold. It should be noted that this design choice has the disadvantage that incorrect first level predictions will lead to second level predictions also being incorrect. However, given that the amount of documents available for training the prediction of the first level categories far exceeds those available for the second level categories, we chose to rely upon the predictions of the first level to guide the second level predictions. The final output of the hierarchical classifier is a one-hot encoded vector which represents the categories that have been assigned to the associated test instance.

4.4 Classifier model architectures

We implemented different classifier model architectures which were used as the multi-class classification models in the hierarchical classifier. These models and techniques include CNNs, attention mechanisms and RNNs.

4.4.1 CNN architectures. We implemented the standard CNN architecture with a CL, PL and a FCL. Furthermore, we experimented with using multiple convolutional filters with different sizes. This allows the CNN to group together different length sequences of word and document embeddings to extract various patterns from the feature vector. We refer to each of these different sized convolutional filters as channels and an architecture with more than one filter size as a multi-channel CNN. Figure 3 presents a multi-channel CNN architecture with three filter sizes of $2 \times s$, $3 \times s$ and $4 \times s$, assuming the S1 fusion strategy is used. The width of the convolutional filters is the dimension of the word embeddings, s, and the heights are chosen to determine the different filter sizes. We use max pooling to extract the maximum value of feature vectors obtained through the CL. We concatenate the results from the PLs associated with the different CL channels to form the input to the FCL. The FCL performs a forward pass to obtain the output value associated with each potential class. Suppose the output of the FCL for instance i is given as $\mathbf{z}_i = [z_{i,1}, z_{i,2}, \ldots, z_{i,K}]$ where K is the number of classes. The FCL then assigns a probability of instance i belonging to class k as $\hat{y}_{i,k}$ by passing the associated output through a softmax function such that

$$\hat{y}_{i,k} = \frac{e^{z_{i,k}}}{\sum_{j=1}^{K} e^{z_{i,j}}} \tag{19}$$

Furthermore, the categorical cross-entropy function is used as the error function and is given by

$$CE = \frac{1}{N} \sum_{i=1}^{N} \left(- \sum_{k=1}^{K} y_{i,k} log(\hat{y}_{i,k}) \right) \tag{20}$$

where N is the number of instances and $y_{i,k}$ is the true output. This error function is used in the backpropagation algorithm to optimise the weights and bias units in the FCL along with the weights of the convolutional filters.

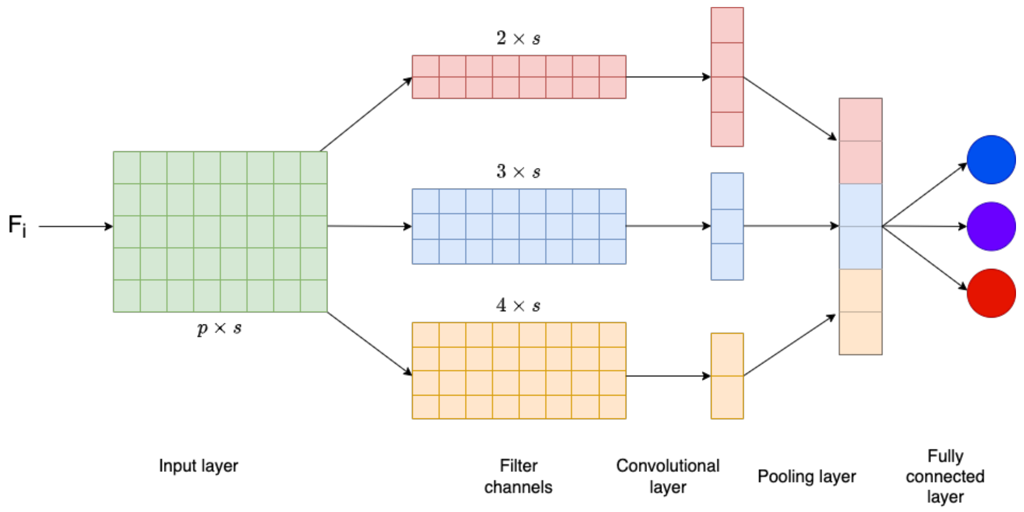


Figure 3. Layers of multi-channel CNN.

4.4.2 Attention mechanisms. We implemented two types of attention mechanisms which are incorporated into the CNN architecture through the replacement of the PL. Thus, these attention mechanisms obtain as input the output of the CL which is a sequence of feature vectors $\mathbf{c}_1, ..., \mathbf{c}_F$ where F is the number of feature vectors and d is the dimension of each vector. For these attention mechanisms we represent the key (**K**) and value (**V**) as the same matrix $\mathbf{C} \in R^{F\times d}$ which is formed by stacking this sequence of feature vectors. The two types of attention mechanisms use different alignment functions and queries and are referred to as label-wise attention and label embedding attention respectively.

The label-wise attention mechanism aims to determine how much attention should be placed on a certain feature when a specific label is being predicted. Thus, the mechanism tries to learn a probability distribution $\boldsymbol{\alpha}^{(k)} \in R^F$ for each label l_k, where the element $\alpha_i^{(k)}$ is a weight which controls the effect that the i-th feature will have when the model is predicting the k-th label. Each query vector in the query matrix, $\mathbf{Q} \in R^{K\times d}$, represents a label, where K is the number of labels and d is the dimension of a feature vector. A dot product alignment function is used to determine the alignment between each query vector and each feature vector to decide where the attention should be placed. The attention weights are calculated as

$$\boldsymbol{\alpha} = \mathrm{softmax}(\mathbf{Q}\mathbf{C}^T + \mathbf{b}) \tag{21}$$

where $\mathbf{Q} \in R^{K\times d}$ and $\mathbf{b} \in R^K$ are randomly initialised and learned during training. We then compute the weighted sum of feature vectors with the attention weights as

$$\mathbf{M} = \boldsymbol{\alpha}\mathbf{C} \tag{22}$$

$\mathbf{M} \in R^{K\times d}$ contains a feature vector for each label such that the output can be determined by applying a softmax activation function as

$$\hat{y}_k = \frac{e^{\mathbf{m}_k \mathbf{w}_k^T + b_k}}{\sum_{i=1}^K e^{\mathbf{m}_i \mathbf{w}_i^T + b_i}} \tag{23}$$

where $\hat{y}_k$ is the output certainty of an instance being assigned to label k, $\mathbf{m}_k$ is the k-th row in $\mathbf{M}$, $\mathbf{w}_k^T$ is the k-th column in $\mathbf{W}$ and b_k is the k-th element in $\mathbf{b}$. $\mathbf{W}$ and $\mathbf{b}$ and randomly initialised and learned during training.

The label embedding attention mechanism introduced by Strydom [42] is an attention mechanism that works similarly to the label-wise attention mechanism but takes advantage of the textual descriptions of categories to guide the classifier into placing attention at the correct locations in the feature vector. The CESM categories each contain a textual description of the category which we want to leverage to improve the performance of the classifier. If we have a document containing the title and abstract of a research output and convert both this document and the textual description of the category the document belongs to through the same embedding method, the intuition is that these representations should be close to one another through a certain distance measure. Thus, the intuition for this attention mechanism is similar to the label-wise attention mechanism but with the aim of exploiting the descriptions of the labels instead of learning these representations from scratch during the training process. Let the i-th input document be represented as a sequence of word tokens given by $S_i = [t_1, \ldots, t_T] \in R^T$. S_i is converted to $D_i = [\mathbf{x}_1, \ldots, \mathbf{x}_T] \in R^{T\times d_e}$ by some embedding method that converts these tokens to word embeddings. This sequence of embeddings, D_i, is then passed through the CNN CLs which results in the feature vectors $\mathbf{c}_1, ..., \mathbf{c}_F$ which are stacked into a matrix $\mathbf{C} \in R^{F\times d}$ where F is the number of feature vectors and d is the dimension of each feature vector.

Suppose we have the text description of the k-th label which is represented by a series of word tokens as L_k =

$[t_1, \ldots, t_M]$ where M is the number of words in the description. We convert the label descriptions using the same embedding mechanism that was used for the input documents. Suppose T is the maximum number of words found in any of the descriptions, descriptions with less than T words are padded with zeros such that the label description for label l_k is represented by the matrix $\mathbf{U}_k \in R^{d_e \times T}$. We learn a vector $\mathbf{p}$ to reduce the dimensionality of this matrix through taking the dot product of the matrix with $\mathbf{p} \in R^T$ to represent the k-th label as a vector,

$$\mathbf{e}_k = \mathbf{U}_k \mathbf{p} \tag{24}$$

We pack these label description embeddings into the query matrix, $\mathbf{Q} = [\mathbf{e}_1, \ldots, \mathbf{e}_K]$. We use this query matrix to perform the general attention mechanism and obtain the attention weights as

$$\boldsymbol{\alpha} = \text{softmax}(\mathbf{QWC}^T) \tag{25}$$

where $\mathbf{W} \in R^{d_e \times d}$ is learned during training.

The final output for this attention mechanism is then determined by using the same procedure as the label-wise attention mechanism.

4.4.3 RNN architecture.

We implemented a LSTM RNN architecture which uses previous hidden states to obtain the final representation of a document as whole as the last hidden state $\mathbf{h}^T$. Furthermore, the use of LSTM cell states uses mechanisms which allows longer sequence information to be remembered by the model such that it does not suffer from the vanishing gradient problem as is the case for standard RNN architectures. The final hidden state is connected to a FCL with K output units where K is the number of classes. The model uses a softmax function to determine the final output for a particular input instance and the categorical cross-entropy error function in the same way as the CNN architectures.

4.4.4 Dropout layer.

We included a dropout layer for each of these model architectures to prevent the model from overfitting the training set. The dropout layer randomly ignores a node and all of its connections with a predefined probability. This introduces some noise into the training procedure and prevents the model from becoming overly reliant on specific connections to nodes in order to improve generalisation performance.

4.4.5 Optimiser.

We used the Adam optimiser [19] which is a combination of the adaptive gradient algorithm (AdaGrad) [13] and root mean square propagation (RMSProp) [37] algorithms. Furthermore, we experiment with different mini-batch sizes which the Adam optimiser uses to perform the backpropagation procedure to find suitable weights and biases for the models.

4.4.6 Hyperparameter tuning.

For each combination of these model architectures and an associated feature extraction method, we performed a hyperparameter tuning process to find the optimal configuration. The general hyperparameters include the number of epochs to train a model, the mini-batch size and the dropout probability. Other hyperparameters include the filter size and the number of filters used in the CNN architectures and the size of the hidden units in the RNN architecture. We used the validation set to measure the performance of a model and determine the configuration which achieves the best generalisation performance. We used a Hyperband tuning algorithm [25] which utilises adaptive resource allocation and early-stopping mechanisms to find good hyperparameters for a model to ensure that it is not overfitting the training set. The algorithm trains a large number of models with random configurations for a few epochs after which only the best performing half of models are considered for the next set of epochs until eventually the optimal configuration with the best performance of the validation set is determined.

4.5 Evaluation

We obtained the optimal configuration for each combination of classifier architecture and feature extraction method through the hyperparameter tuning process and compared their performance on the test set. The test set contains 1488 text documents with their associated CESM category assignments. We evaluated the performance of a particular model by considering the top two CESM category levels and using the precision (P), recall (R), F1 score (F1) and subset accuracy (SA) performance measures. Due to the stochastic training process of these models, we obtained the average over 10 independent runs as the measure of performance such that they can be reliably compared. Furthermore, we used the standard deviation over the 10 independent runs to determine the stability of these models.

Consider a true positive (TP) and a false positive (FP) as a correctly and incorrectly assigned category respectively. Similarly, consider a true negative (TN) and a false negative (FN) as a correctly and incorrectly unassigned category respectively. We represent the total number of TP, FP, TN and FN assignments as TP_{total}, FP_{total}, TN_{total} and FN_{total} respectively.

P is defined as the proportion of correctly assigned labels to the total number of assigned labels, *i.e.* $\text{TP}_{\text{total}}/(\text{TP}_{\text{total}} + \text{FP}_{\text{total}})$. Thus, it measures how accurate a positively assigned label is on average and is calculated as

$$\text{P} = \frac{1}{N} \sum_{i=1}^{N} \frac{|\mathbf{y}_i \cap \hat{\mathbf{y}}_i|}{|\hat{\mathbf{y}}_i|} \tag{26}$$

where N is the number of test samples and $\mathbf{y}_i$ and $\hat{\mathbf{y}}_i$ denote the true and predicted labels for document i respectively.

R is defined as the proportion of correctly predicted labels to the total number of true labels, *i.e.* $\text{TP}_{\text{total}}/(\text{TP}_{\text{total}}+\text{FN}_{\text{total}})$. Thus, it measures how many of the positive label assignments the model captures on average and is calculated as

$$\mathrm{R} = \frac{1}{N}\sum_{i=1}^{N}\frac{|\mathbf{y}_i \cap \hat{\mathbf{y}}_i|}{|\mathbf{y}_i|} \tag{27}$$

F1 combines P and R to provide a comprehensive measure of the performance, where higher values indicate better performance, and is given by

$$\mathrm{F1} = \frac{2P \times R}{P + R} \tag{28}$$

SA evaluates the fraction of samples that are classified exactly correct and is given by

$$\mathrm{SA} = \frac{1}{N}\sum_{i=1}^{N} I(\mathbf{y}_i = \hat{\mathbf{y}}_i) \tag{29}$$

where $I(\mathbf{y}_i = \hat{\mathbf{y}}_i)$ is 1 if all labels are correctly predicted and 0 otherwise.

5 Results

We evaluated each of our models using the test set to determine which model performs the best in general for the task of hierarchically classifying research documents. We considered the different types of models as the standard CNN, multi-channel CNN, label-wise attention CNN, label embedding attention CNN and the RNN. For each of these models we compared each of the feature extraction methods which includes the benchmark word embedding technique, S0, and the three fusion strategies, S1, S2 and S3. We obtained the performance of these models as the average over 10 independent runs and use the standard deviation over these runs to determine the stability of the models. These results were obtained by using the optimal configuration for each model and feature extraction method combination as determined by the hyperparameter tuning process described in Section 4.4.6.

5.1 Standard CNN results

Table 2 presents the results obtained with the standard CNN architecture for each of the different feature extraction methods. We observed that the S3 feature extraction method outperformed the other methods for each of the performance measures and obtained the highest F1 score (0.6676) which narrowly outperformed S2 (0.6644). All three of the fusion strategies outperformed S0 by a significant margin for each of the performance measures. Furthermore, we observed that the S1 and S2 strategies provide different trade-offs, with S1 obtaining higher P and SA scores and S2 obtaining higher R and F1 scores.

Table 2. Standard CNN results.

Feature extraction	P	R	F1	SA
S0	0.6001	0.6163	0.6081	0.3958
S1	0.6505	0.6683	0.6593	0.4698
S2	0.6488	0.6808	0.6644	0.4594
S3	0.6511	0.6849	0.6676	0.4758

5.2 Multi-channel CNN results

Table 3 provides the results obtained with the multi-channel CNN for the different feature extraction methods. From Table 3 we observed that the highest F1 score was obtained by S2 (0.6768), followed by S3 (0.6733), S1 (0.6662) and S0 (0.6461). However, S2 and S3 obtained very similar results for each of the performance measures, with S3 outperforming S2 in terms of SA with scores of 0.4871 and 0.4813 respectively and S2 obtaining higher scores for the other performance measures. We observed that the precision obtained by S0 (0.6683) was higher than S1 (0.6605) but that the recall obtained by S0 (0.6257) was significantly lower than S1 (0.6724).

Table 3. Multi-channel CNN results.

Feature extraction	P	R	F1	SA
S0	0.6683	0.6257	0.6461	0.4149
S1	0.6605	0.6724	0.6662	0.4641
S2	0.6757	0.6759	0.6768	0.4813
S3	0.6756	0.6710	0.6733	0.4871

5.3 Label-wise attention CNN results

Table 4 shows the results obtained with the label-wise attention CNN for each of the different feature extraction methods. We observed that, with regard to the F1 score, S0 (0.5209) performed significantly worse than S1 (0.6566), S2 (0.6345) and S3 (0.6376). From Table 4 we observed that S1 outperformed the other feature extraction methods for each of the performance measures. The S2 and S3 fusion strategies obtained a very similar level of performance for each of the performance measures.

Table 4. Label-wise attention CNN results.

Feature extraction	P	R	F1	SA
S0	0.5813	0.4722	0.5209	0.2706
S1	0.6525	0.6609	0.6566	0.4578
S2	0.6319	0.6372	0.6345	0.4457
S3	0.6284	0.6472	0.6376	0.4399

5.4 Label embedding attention CNN results

Table 5 presents the results obtained with the label embedding attention CNN for the different feature extraction methods. We observed that S1 outperformed the other fusion strategies for each of the performance measures. Furthermore, we observed that the F1 scores obtained by S0 (0.5837) and S3 (0.5853) were very similar and lower than the score obtained by S2 (0.6177).

Table 5. Label embedding attention CNN results.

Feature extraction	P	R	F1	SA
S0	0.6235	0.5488	0.5837	0.2552
S1	0.6265	0.6392	0.6328	0.3786
S2	0.6186	0.6169	0.6177	0.3389
S3	0.5787	0.5921	0.5853	0.3383

5.5 RNN results

Table 6 provides the results obtained with the LSTM RNN for the different feature extraction methods. We observed that S3 outperformed each of the other fusion strategies in terms of precision (0.6257), F1 score (0.6049), and subset accuracy (0.3919). Furthermore, the F1 scores obtained by S1 (0.5870), S2 (0.5847), and S0 (0.5734) were very similar.

Table 6. RNN results.

Feature extraction	P	R	F1	SA
S0	0.5430	0.6075	0.5734	0.3734
S1	0.5856	0.5896	0.5870	0.3918
S2	0.5614	0.6099	0.5847	0.3875
S3	0.6257	0.5857	0.6049	0.3919

5.6 Model stability

To measure the stability of the different model architectures and feature extraction methods, we used the standard deviation of the F1 score obtained on the test set over 10 independent runs. Table 7 presents the standard deviation of the F1 scores for each of the combinations. From Table 7 we observed that the lowest standard deviation of 0.0020 was obtained by the standard CNN with the S0 feature extraction strategy followed by 0.0021 obtained by the standard CNN with the S2 feature extraction strategy and 0.0028 obtained by the multi-channel CNN with the S2 feature extraction strategy. Furthemore, we observed that the highest standard deviations were obtained by the label embedding attention CNN with the S2 (0.0157) and S1 (0.0156) feature extraction strategies.

Table 7. Standard deviation of F1 scores.

	S0	S1	S2	S3
Standard CNN	0.0020	0.0118	0.0021	0.0087
Multi CNN	0.0090	0.0090	0.0028	0.0033
Label-wise attention	0.0128	0.0056	0.0040	0.0061
Embedding attention	0.0046	0.0156	0.0157	0.0069
RNN	0.0040	0.0122	0.0081	0.0104

5.7 Discussion

We considered four different feature extraction strategies along with five different classification model architectures to determine which combination obtains the best results for the classification of South African research outputs. From the results obtained by the experiments we found that the multi-channel CNN with the S2 fusion strategy obtained the best results in terms of precision, recall and F1 score. However, the same classification model obtained very similar results for the S3 fusion strategy which achieved the best performance in terms of subset accuracy. We hypothesise that the S2 and S3 fusion strategies outperform the other feature extraction methods due to the combination of global topic and local semantic information provided by the LDA and SciBERT models respectively. This combination of information may provide a better semantic representation of the document by considering both the topic information of the corpus as a whole and the local contextual semantics of the words in a document. This mixture of different information can then be used to more accurately categorise the associated document.

The multi-channel CNN outperformed the standard CNN architecture for each of the feature extraction strategies, albeit not by a large margin. We hypothesise that this increase in performance can be attributed to the ability of the multi-channel CNN to use different filter sizes which groups varying lengths of words together to extract meaningful patterns which can not always be formed by a single filter size. Furthermore, the general performance trend indicated that the standard CNN obtained the second best results, followed by the label-wise attention CNN and the embedding attention CNN, with the LSTM RNN classifier obtaining the worst results in general.

We observed that the label-wise attention CNN outperformed the label embedding CNN for each of the feature extraction strategies apart from S0. We hypothesise that the worse performance of the label embedding attention CNN can be attributed to the category descriptions in the CESM documentation often not being similar to the abstracts of the associated categories. Thus, the label-wise attention which effectively attempts to learn a representation for a particular category from random initialisation can achieve better results as it is not hindered by inaccurate descriptions.

In term of the stability of the models over the execution of independent runs we observed that the multi-channel CNN obtained relatively low standard deviations for each of the fusion strategies compared to the other classification models and we thus hypothesise that it is the most stable model in general for the associated task with the given feature extraction methods.

6 Future work

In future work, we plan to apply the developed hierarchical classifiers to more domains and datasets to obtain a generalised measure of how these classifiers perform. Potential domains include the hierarchical classification of books, movies, songs, and medical notes to assist doctors with the diagnosis of illnesses.

We will experiment with the use of different pre-trained transformer-based models, such as RoBERTa [29] and ALBERT [22], to compare the performance of these models as feature extraction methods for text classification. Furthermore, we will fine-tune these pre-trained models for the task of text classification and use the fine-tuned model to obtain the semantic feature representation of documents.

We will develop more classification models by extending the architecture of the developed models and considering different architectures such as transformer models. Furthermore, we will investigate different hierarchical classification architectures which exploit the hierarchical structure of categories in different ways to obtain more accurate predictions.

7 Conclusion

The objective of this project was to develop an approach to categorise South African research outputs in a hierarchical structure according to the official CESM standard. To solve this problem we developed a hierarchical text classifier which uses the title and abstract of a master's thesis or PhD dissertation and assigns CESM categories to that document.

We used topic models and transformer-based language models to obtain global topic and local semantic representations of documents respectively. We used different fusion mechanisms to combine the information obtained from these two models to form the input to a classification model. We implemented five different classification architectures which included a standard CNN, multi-channel CNN, two attention mechanisms as extensions to the standard CNN, and a LSTM RNN. Finally, we performed experiments with each of the classification models and feature extraction methods to obtain the combination which is best suited for the task of hierarchically classifying these text documents.

We found that the multi-channel CNN outperformed the other classification model architectures, with the highest F1 scores (0.6461, 0.6662, 0.6768, 0.6733) for each of the four feature extraction methods. It also obtained the best performance for the feature extraction methods which combine the information obtained by the LDA and SciBERT models. These feature extraction methods (S2 and S3) combine the global topic and local context semantic representations obtained by two models to improve classification performance on the given dataset.

References

[1] M. Ansari and N. M. Shahane. 2018. A review on multi-label classification. *International Journal of Research and Analytical Reviews* 6, 1 (December 2018), 816–819.

[2] D. Bahdanau, K. Cho, and Y. Bengio. 2014. Neural machine translation by jointly learning to align and translate. https://doi.org/10.48550/ARXIV.1409.0473

[3] I. Beltagy, K. Lo, and A. Cohan. 2019. SciBERT: A pretrained language model for scientific text. Association for Computational Linguistics. https://www.aclweb.org/anthology/D19-1371

[4] D.M. Blei. 2012. Probabilistic topic models. *ACM* 55, 4 (April 2012), 77–84. https://doi.org/10.1145/2133806.2133826

[5] D.M. Blei, A.Y. Ng, and M.I. Jordan. 2003. Latent Dirichlet Allocation. *Journal of Machine Learning Research* 3 (March 2003), 993–1022.

[6] J. Bogatinovski, L. Todorovski, S. Džeroski, and D. Kocev. 2022. Comprehensive comparative study of multi-label classification methods. *Expert Systems with Applications* 203 (April 2022), 117–215. https://doi.org/10.1016/j.eswa.2022.117215

[7] M. Boutell, J. Luo, X. Shen, and C. Brown. 2004. Learning multi-label scene classification. *Pattern Recognition* 37 (March 2004), 1757–1771. https://doi.org/10.1016/j.patcog.2004.03.009

[8] N. Cesa-Bianchi, C. Gentile, and L. Zaniboni. 2006. Incremental algorithms for hierarchical classification. *Journal of Machine Learning Research* 7 (December 2006), 31–54.

[9] G. Chen, D. Ye, Z. Xing, J. Chen, and E. Cambria. 2017. Ensemble application of convolutional and recurrent neural networks for multi-label text categorization. In *Proceedings of the 2017 International Joint Conference on Neural Networks*. 2377–2383. https://doi.org/10.1109/IJCNN.2017.7966144

[10] F. Chollet. 2015. *Keras*. https://github.com/fchollet/keras

[11] K. Daphne and S. Mehran. 1997. Hierarchically classifying documents using very few words. In *International Council for Machinery Lubrication.*

[12] J. Devlin, M. Chang, K. Lee, and K. Toutanova. 2019. 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)*. Association for Computational Linguistics, Minneapolis, Minnesota, 4171–4186. https://doi.org/10.18653/v1/N19-1423

[13] J. Duchi, E. Hazan, and Y. Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research* 12 (July 2011), 2121–2159.

[14] R. Eugenia. 1986. An algorithm for finding nearest neighbours in (approximately) constant average time. *Pattern Recognition Letters* 4 (October 1986), 145–157.

[15] C. J. Fall, A. Törcsvári, K. Benzineb, and G. Karetka. 2003. Automated categorization in the international patent classification. *Special Interest Group on Information Retrieval Forum* 37, 1 (April 2003), 10–25. https://doi.org/10.1145/945546.945547

[16] S. Hochreiter and J. Schmidhuber. 1997. Long short-term memory. *Neural computation* 9 (December 1997), 1735–80. https://doi.org/10.1162/neco.1997.9.8.1735

[17] T. Hofmann. 2013. Probabilistic latent semantic analysis. https://doi.org/10.48550/ARXIV.1301.6705

[18] Y. Kim. 2014. Convolutional neural networks for sentence classification. *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*. https://doi.org/10.3115/v1/D14-1181

[19] D. P. Kingma and J. Ba. 2014. Adam: A method for stochastic optimization. https://doi.org/10.48550/ARXIV.1412.6980

[20] S. Kiritchenko, R. Nock, and F. Famili. 2006. Learning and evaluation in the presence of class hierarchies: application to text categorization. *Advances in Artificial Intelligence* 4013 (June 2006), 395–406. https://doi.org/10.1007/11766247_34

[21] Y. Labrou and T. Finin. 1999. Yahoo! As an ontology: using Yahoo! categories to describe documents. 180–187. https://doi.org/10.1145/319950.319976

[22] Z. Lan, M. Chen, S. Goodman, K. Gimpel, P. Sharma, and R. Soricut. 2019. ALBERT: A lite BERT for self-supervised learning of language representations. https://doi.org/10.48550/ARXIV.1909.11942

[23] Y. Lecun and Y. Bengio. 1995. Convolutional Networks for Images, Speech, and Time-Series. *The Handbook of Brain Theory and Neural Networks*, 255–258.

[24] F. Li and H. Yu. 2019. ICD coding from clinical text using multi-filter residual convolutional neural network. https://doi.org/10.48550/ARXIV.1912.00862

[25] L. Li, K. Jamieson, G. DeSalvo, A. Rostamizadeh, and A. Talwalkar. 2016. Hyperband: A novel bandit-based approach to hyperparameter optimization. (2016). https://doi.org/10.48550/ARXIV.1603.06560

[26] S. Linnainmaa. 1976. Taylor expansion of the accumulated rounding error. *BIT* 16, 2 (June 1976), 146–160. https://doi.org/10.1007/BF01931367

[27] J. Liu, Wei-C. Chang, Y. Wu, and Y. Yang. 2017. Deep learning for extreme multi-label text classification. Association for Computing Machinery, New York, NY, USA. https://doi.org/10.1145/3077136.3080834

[28] W. Liu, J. Pang, N. Li, X. Zhou, and F. Yue. 2021. Research on multi-label text classification method based on tALBERT-CNN. *International Journal of Computational Intelligence Systems* 14, 1 (January 2021). https://europepmc.org/articles/PMC8666839

[29] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov. 2019. RoBERTa: A robustly optimized BERT pretraining approach. https://doi.org/10.48550/ARXIV.1907.11692

[30] E.R. Miguel and S. Padmini. 2004. Hierarchical text categorization using neural networks. *Information Retrieval* 5 (2004), 87–118.

[31] J. Mullenbach, S. Wiegreffe, J. Duke, J. Sun, and J. Eisenstein. 2018. Explainable prediction of medical codes from clinical text. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*. Association for Computational Linguistics, New Orleans, Louisiana, 1101–1111. https://doi.org/10.18653/v1/N18-1100

[32] C. Nwankpa, W. Ijomah, A. Gachagan, and S. Marshall. 2018. Activation Functions: comparison of trends in practice and research for deep learning. https://doi.org/10.48550/ARXIV.1811.03378

[33] T. O'Malley, E. Bursztein, J. Long, F. Chollet, H. Jin, and L. Invernizzi. 2019. Keras Tuner. https://github.com/keras-team/keras-tuner.

[34] N. Peinelt, D. Nguyen, and M. Liakata. 2020. tBERT: Topic models and BERT joining forces for semantic similarity detection. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, Online, 7047–7055. https://doi.org/10.18653/v1/2020.acl-main.630

[35] A. Radford and K. Narasimhan. 2018. Improving language understanding by generative pre-training.

[36] R. Rehurek and P. Sojka. 2011. Gensim-python framework for vector space modelling. *NLP Centre, Faculty of Informatics, Masaryk University, Brno, Czech Republic* 3, 2 (2011).

[37] S. Ruder. 2016. An overview of gradient descent optimization algorithms. https://doi.org/10.48550/ARXIV.1609.04747

[38] D.E. Rumelhart, G.E. Hinton, and R.J. Williams. 1986. Learning internal representations by error propagation.

[39] G. Salton. 1971. *The SMART retrieval system—experiments in automatic document processing*. Prentice-Hall, Inc., USA.

[40] M.S. Sorower. 2010. A literature survey on algorithms for multi-label learning.

[41] R.A. Stein, P.A. Jaques, and J.F. Valiati. 2019. An analysis of hierarchical text classification using word embeddings. *Information Sciences* 471 (2019), 216–232.

[42] S. Strydom. 2021. *Automatic assignment of diagnosis codes to free-form text medical notes*. Master's thesis. Stellenbosch University.

[43] A. Vani, Y. Jernite, and D. Sontag. 2017. Grounded recurrent neural networks. https://doi.org/10.48550/ARXIV.1705.08557

[44] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N Gomez, L. Kaiser, and I. Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*, Vol. 30. Curran Associates, Inc. https://proceedings.neurips.cc/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf

[45] T. Vu, D.Q. Nguyen, and A. Nguyen. 2020. A label attention model for ICD coding from clinical text. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence*. https://doi.org/10.24963/ijcai.2020/461

[46] T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz, J. Davison, S. Shleifer, P. von Platen, C. Ma, Y. Jernite, J. Plu, C. Xu, T.L. Scao, S. Gugger, M. Drame, Q. Lhoest, and A.M. Rush. 2019. HuggingFace's Transformers: State-of-the-art natural language processing. https://doi.org/10.48550/ARXIV.1910.03771

[47] R. Yarullin and P. Serdyukov. 2021. *BERT for sequence-to-sequence multi-label text classification*. 187–198. https://doi.org/10.1007/978-3-030-72610-2_14

[48] S.H. Zellig. 1954. Distributional structure. *WORD* 10, 2-3 (1954), 146–162. https://doi.org/10.1080/00437956.1954.11659520

[49] M. Zhang and Z. Zhou. 2014. A review on multi-label learning algorithms. *Knowledge and Data Engineering, IEEE Transactions on* 26 (August 2014), 1819–1837. https://doi.org/10.1109/TKDE.2013.39

A Implementation

We used two libraries to obtain the word and document embeddings using the LDA and SciBERT models. To obtain the LDA information we used the Gensim [36] library which provides the resulting distributions of documents over topics and topics over words. From these distributions we determined the document and word embeddings. To obtain the SciBERT word and document embeddings we used the Hugging Face [46] library which tokenises the input documents and provides the resulting embeddings. For our implementation of the various machine learning models we used the Tensorflow and Keras [10] libraries. The libraries have built-in implementations of the standard CNN and RNN architectures used in this project. For the implemented attention mechanisms we developed custom Keras layers. To tune these models we used the Keras Tuner [33] library.

Final lines-of-code count: 3200