FACTA UNIVERSITATIS
Series: **Electronics and Energetics** Vol. 37, Nº 4, December 2024, pp. 703 – 732
https://doi.org/10.2298/FUEE2404703J

Original scientific paper

HIERARCHICAL TEXT CLASSIFICATION FOR WEB OF SCIENCE SCIENTIFIC FIELDS*

Pouyan Jahani Rad, Mahdi Bahaghighat

Computer Engineering Department, Imam Khomeini International University, Qazvin, Iran

ORCID iDs: Pouyan Jahani Rad https://orcid.org/0009-0007-2956-2209
Mahdi Bahaghighat https://orcid.org/0000-0002-1813-8417

Abstract. *This research focuses on making an efficient text classifier to map given corpora to specific scientific fields. Our study is set on the classification of different scientific fields according to the categories of the Web of Science (WOS). We design and develop various deep learning architectures such as Convolutional Neural Network (CNN), Deep Neural Network (DNN), and Recurrent Neural Network (RNN) at both the parent and child levels. To make our model perform better, we effectively use Hyperband Tuning. We aim to construct a precise hierarchical text classifier for lower levels, and smaller general model sizes. The evaluation employs a special metric known as the hierarchical confusion matrix. Results based on a broad investigation of Word Embedding, Document Embedding, and Hyperband Tuning show that the hierarchical combination of CNN and DNN in parent-child levels can achieve greater accuracy. Our model scored genuinely well, with an F1 score of 94.29% and an accuracy of 99.33%. Although using an RNN at the parent level and another at the child level led to lower accuracy, it effectively reduced the overall model size. We also conducted a comprehensive evaluation of various model architectures using the AoI2WoS dataset. By incorporating Google News word embeddings, we tested different combinations of RNN-DNN and RNN-RNN models on the AoI2WoS dataset. The RNN-DNN model yielded the best results, achieving an accuracy of 98.71% and an F1-score of 91.87%*. *These findings not only push forward the development of hierarchical text classification but also provide potent tools for utilizing research in scientometrics, and bibliometric researches.*

Key words: *Hierarchical text classification, deep learning, scientometrics, Google Scholar, WOS.*

Received May 23, 2024; revised September 08, 2024 and October 13, 2024; accepted November 01, 2024
Corresponding author: Mahdi Bahaghighat
Computer Engineering Department, Imam Khomeini International University, Qazvin, Iran
E-mail: Bahaghighat@eng.ikiu.ac.ir

1. INTRODUCTION

In recent years, artificial intelligence (AI) and machine learning (ML) have rapidly improved. Natural Language Processing (NLP) is an essential area of machine learning that studies how computers understand human language. It enables machines to recognize, analyze, and create text consisting of humans. It is a game changer for many applications relevant to human-computer interaction, chatbots, sentiment analysis, information mining systems, and communication. Text mining and classification are important aspects of NLP, helping to make sense of unstructured text data by converting it into structured insights and knowledge. These are all about sifting through enormous quantities of text to extract applicable records. They help us navigate and understand the vast amount of written material available [1-5].Text classification (TC) is a vital field in NLP focusing on classifying text documents into predefined classes. It is crucial to many applications, including topic classification, information retrieval, sentiment analysis, spam detection, etc. [6]. It can methodically examine large volumes of text data, find information, and provide customized user experiences. The need for accurate and efficient TC algorithms has become increasingly significant because of the exponential growth of digital content.

In hierarchical classification, the objective is to organize data in a hierarchy of classes and subcategories [7, 8]. It aids in more accurate classification, especially in areas where data naturally form a hierarchy, such as taxonomies, by making the information more accessible and recognizable, it helps manage more complex tasks and provides flexibility in handling a wide range of classification types [9]. Due to the vast amount of information available today and the advantages of effectively processing hierarchical data, these capabilities have become increasingly important [10]. By identifying the relationships among classes within the hierarchical structure, classification methods become more efficient in achieving accurate results.

On the other side, NLP can revolutionize scientometrics by enabling more efficient and meaningful analysis of scientific literature, enhancing metrics of research impact, and improving collaboration and innovation in science in particular for examining large amounts of data. NLP techniques are highly impactful, enabling tasks such as automating literature reviews, analyzing citation networks, identifying research trends, and forecasting future research trajectories in both scientometrics and bibliometrics. Integration of ML/NLP with bibliometric techniques strengthens the evaluation of research outputs, enhances knowledge discovery, and supports evidence-based decision-making, thereby driving innovation and efficiency in scholarly communication and academic analysis, and scientometric researchers can save time and reduce costs with AI-powered models [11, 12].

In this study, we aimed to develop a high-performing text classifier that categorizes academic texts into specific academic fields according to Web of Science (WOS). Our approach utilizes deep learning techniques to simplify the tedious and costly process of classifying academic information into predetermined scientific disciplines by providing a helpful tool for automatic categorization. Potential benefits include facilitating literature searches, fostering collaboration among researchers with related interests, providing information for resource allocation, and organizational strategic planning. In addition, this type of classifier supports search and advice systems in academic databases and helps pick out new scientific trends. This article presents a multifaceted exploration of hierarchical text classification, notably comparing diverse pre-trained word embeddings like Glove, Word2Vec, and Fast Text on the WOS categories. The primary goal is to strategically implement a variety of reliable

CNN, DNN, and RNN architectures to achieve accurate results, as measured by the hierarchical confusion matrix. The study aims to identify the best-performing word embedding through hierarchical deep-learning models. A parallel investigation into document embedding techniques underscores the efficacy of Word2Vec in hierarchical classification. The research also conducts a meticulous comparative analysis between word, and document embeddings, highlighting the superior accuracy of the former in hierarchical text classification. With the help of our WOS fields' classification method and other advanced techniques of data analysis, such as machine learning, network analysis, and text mining, it is possible to gather insights into the structure, dynamics, and impact of scientific research in different fields for aided informed decision-making for scientific advancement. In this study, we also explored the performance of different model architectures using our public dataset that entitled AoI2WoS. The AoI2WoS dataset played a crucial role in evaluating the capabilities of these models, allowing us to draw meaningful conclusions about their accuracy and efficiency. By applying Google News word embeddings, we tested combinations of RNN-DNN and RNN-RNN models to assess their effectiveness. Our findings underscore the importance of selecting appropriate model architectures for optimal performance on this dataset.

The rest of the paper is organized as follows: First, we look at Related Works, discussing present research. This section offers a summary of what different researchers have achieved and the way they have contributed. Then, in the Methodology section, we explain different deep learning models and how we use them for hierarchical text classification. After that, in Experiment and Results, we explain how we performed our experiments and what we determined. Finally, in Conclusion and Future Works, we summarize the key findings of the study and discuss what future research should explore.

2. Related Work

Notable progress has been made in NLP recently, mainly in word embeddings. Researchers have looked into combining different word embeddings to make them better at understanding language and doing well in tasks. To address the demanding situations of excessive multi-class, and multi-label textual content classification [13] introduced a model called Hierarchical Label Set Expansion (HLSE). It tests how using one-of-a-kind ways to recognize words and sentences impacts its performance consisting of the use of unique methods to symbolize words, and taking note of both grammar and sentence structure.

Document embedding is a subset of feature extraction that uses word embeddings to capture words or texts rather than individual words. In [14], the aim was to generate unsupervised word or document embeddings, going beyond the effective Word2Vec method for single words. The Tens-Embedding application enhanced text classification, sentence-level, and document-level sentiment analysis, as well as text clustering tasks. Experiments on the 20 newsgroups, R52, R8, MR, and IMDB datasets demonstrated the proposed method's superiority over other document embedding techniques.

In [13], they addressed the issue by unveiling a new approach to report embedding, ingeniously combining subjects and sentiments to enhance marketplace forecasting. This progressive embedding approach was jointly trained alongside financial time series information within a device studying framework, and its effectiveness was classified via obligations aimed at predicting marketplace traits. It refers to recent tries to compress learned representations using linear or non-linear strategies, highlighting studies guidelines in memory-green dense representations [15]. These methodologies had been applied to a hierarchical textual content, correctly addressing the demanding situations of multi-label category troubles [16].

In [17], it is emphasized that optimizing hyperparameters is crucial because they determine how well the predictive model performs. Because of the incomplete understanding of the complex relationship between hyperparameters and how they affect model quality, they concluded the necessity of resorting to brute-force or manual tuning techniques. Bayesian optimization techniques have turned out to be an excellent choice that specializes in quickly locating pleasant the tedious and costly process of classifying academic information into predetermined scientific disciplines by providing a helpful tool for automatic categorization. While proof showed they were better than random search, those methods conflict with the challenge of tweaking, and refining functions with many variables and could have a few random noises. Many Bayesian optimization techniques should be performed one step at a time which makes it more difficult to guarantee consistent consequences.

A decrease in performance has been observed in traditional supervised classifiers as the number of documents and categories has increased [18, 19]. In [20], they introduced a new classification model called HDLTex. It takes a one-of-a-kind approach, searching at classification as a hierarchy instead of a massive organization of classes. HDLTex uses deep learning at each level of the hierarchy to improve understanding. These hierarchical classification approaches have a wide range of applications, like figuring out sentiment, labeling topics, and spam detection [21].

3. Methodology

Today, scientometrics can efficiently use NLP and text classification to automate the classification of academic publications into different disciplines, discover research interests, and extract keywords from scientific articles. This equipment facilitates collaboration evaluation and permits researchers to become aware of styles of co-authorship and affiliation. Textual content categorization and NLP aid scientometric research by automating literature searches, improving information retrieval, and imparting valuable insights into the dynamics, patterns, and shapes of the scientific community [22].

Hierarchical classification is used for situations where categories are organized in a hierarchy. This approach provides a clearer view of how different categories are related. When dealing with documents that fit into a couple of classes, the hierarchical type allows us to prepare subjects in a structured way. For example, we might start by sorting abstracts into large classes like "Computer", and "Medical," then split the ones down into smaller groups like "Computer Vision" or "Machine Learning," as you may see in Fig. 1.

This article presents a multifaceted exploration of hierarchical text classification, namely on the WOS categories by comparing diverse word embedding techniques like Glove, Word2Vec, and Fast Text. The main concern is to perform the strategic implementation of a bunch of reliable CNN, DNN, and RNN architectures to get accurate results as evaluated by the hierarchical confusion matrix.

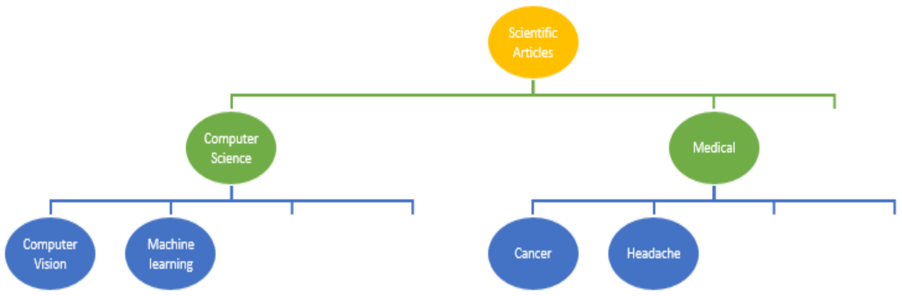


Fig. 1 A partial view of hierarchical classes in the WOS46985 dataset [20]

Through the lens of hierarchical deep learning models, the study tries to explore the superior word embedding performer. Table 1 outlines diverse proposed scenarios in our study, each presenting a unique description and corresponding results. Notably, the outcomes of these scenarios are discussed and compared in the following sections.

Table 1 Different suggested scenarios with related tasks in this study

Item	S1	S2	S3	S4	S5	S6
Word Embedding (WE)	✓	✓	☒	☒	☒	✓
WE Dimension	✓		☒	☒	☒	✓
Document Embedding (DE)	☒	☒	✓	☒	☒	✓
DE (Avg, Max, Min) D	☒	☒	☒	✓	☒	✓
DE (Avg, Max, Min)	☒	☒	☒	✓	✓	✓
Hyperband Tunning	✓	✓	✓	✓	✓	✓

In the first level of our architecture, we employed a CNN comprising two convolutional layers and corresponding max-pooling layers. This initial stage of processing allows the network to learn hierarchical representations from the input data. Following the convolutional and pooling layers, we introduced a flattened layer to reshape the output into a one-dimensional array, preparing it for further processing. Moving forward in our neural network design, we incorporated a dense layer with 200 neurons. This layer serves as a feature extractor, capturing complex patterns and relationships within the data learned during the convolutional stages. The utilization of 200 neurons enables the model to encode a rich set of features, enhancing its ability to understand and represent intricate patterns in the input.

To finalize our architecture, we added a second dense layer with 7 neurons, corresponding to the desired output dimension. This layer, with its 7 neurons, is tailored to the specific task at hand, effectively transforming the learned features into meaningful predictions or classifications. The choice of 7 neurons aligns with the output requirements because we have 7 labels for classification in Fig. 2.

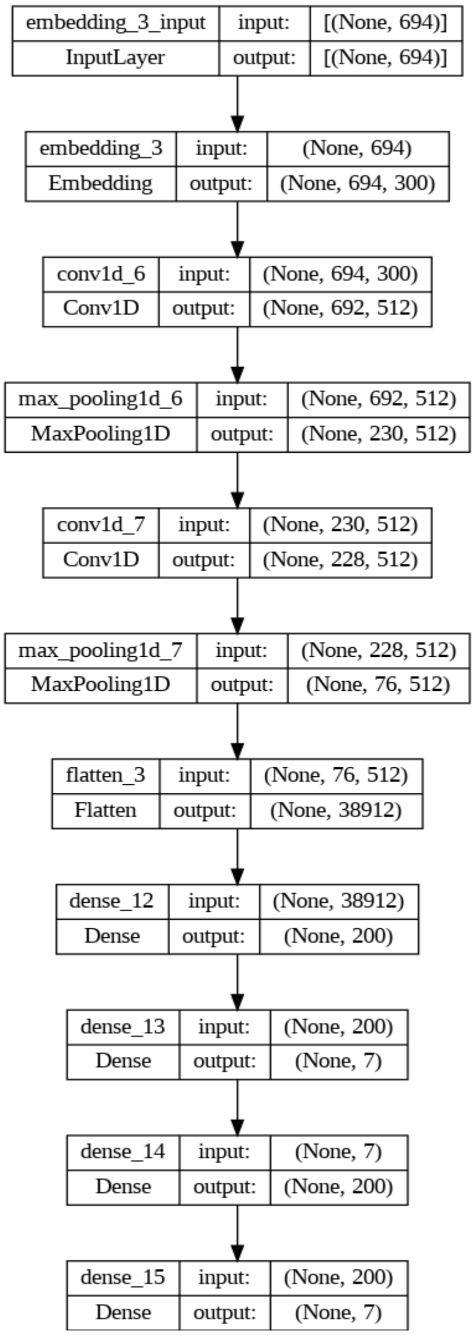

embedding_3_input | input: | [(None, 694)]
InputLayer | output: | [(None, 694)]
embedding_3 | input: | (None, 694)
Embedding | output: | (None, 694, 300)
conv1d_6 | input: | (None, 694, 300)
Conv1D | output: | (None, 692, 512)
max_pooling1d_6 | input: | (None, 692, 512)
MaxPooling1D | output: | (None, 230, 512)
conv1d_7 | input: | (None, 230, 512)
Conv1D | output: | (None, 228, 512)
max_pooling1d_7 | input: | (None, 228, 512)
MaxPooling1D | output: | (None, 76, 512)
flatten_3 | input: | (None, 76, 512)
Flatten | output: | (None, 38912)
dense_12 | input: | (None, 38912)
Dense | output: | (None, 200)
dense_13 | input: | (None, 200)
Dense | output: | (None, 7)
dense_14 | input: | (None, 7)
Dense | output: | (None, 200)
dense_15 | input: | (None, 200)
Dense | output: | (None, 7)

Fig. 2 The proposed model for Parent-level architecture

In the second level of our architecture, we transitioned from convolutional layers to a DNN design. This shift allows us to refine the features extracted by the CNN in the first level, capturing more intricate relationships and patterns in the data. Within the DNN architecture, we strategically incorporated dense layers to enhance the network's capacity for learning complex representations. Following each dense layer, a dropout layer was introduced as a regularization technique to mitigate the risk of overfitting. Dropout layers randomly deactivate a specified percentage of neurons during training, preventing the model from relying too heavily on specific features and promoting a more robust and generalized learning process to address the diverse outputs required for different areas of interest, we employed separate dense layers after the dropout layers. Each of the dense layers are tailored to the specific characteristics and nuances of the output associated with different regions or categories, as illustrated in Fig. 3. The overall architecture, transitioning from CNNs to DNNs in a multi-level fashion, reflects a hierarchical learning process. The convolutional layers excel at capturing spatial hierarchies and patterns, while the subsequent DNN layers specialize in abstracting and refining these features for the diverse outputs associated with different areas or categories. The combination of dropout regularization and specialized dense layers contributes to a robust and adaptable model capable of handling complex tasks across various domains.

3.1. Dataset

In this study, we used the WOS46985 [20] public dataset along with our dataset named the AoI2WoS [23].

3.1.1. WOS46985 dataset

At the first, a public dataset called the WOS46985 dataset [20] was used in our study. It contains 46985 documents with four fields. We considered three fields including {Abstract, Domain, Area} for the experiment. The abstract text is used for model input. Table 2 lists the following disciplines in the Domain field: computer science, electrical engineering, mechanical engineering, psychology, civil engineering, medical science, and biochemistry.

Table 2 Number of documents for each domain in the WOS46985 dataset [20]

Domain	No. Areas
Computer Science	6514
Electrical Engineering	5483
Medical Sciences	14625
Psychology	7142
Biochemistry	5687
Civil Engineering	4237
Mechanical Engineering	3297
Total	46985

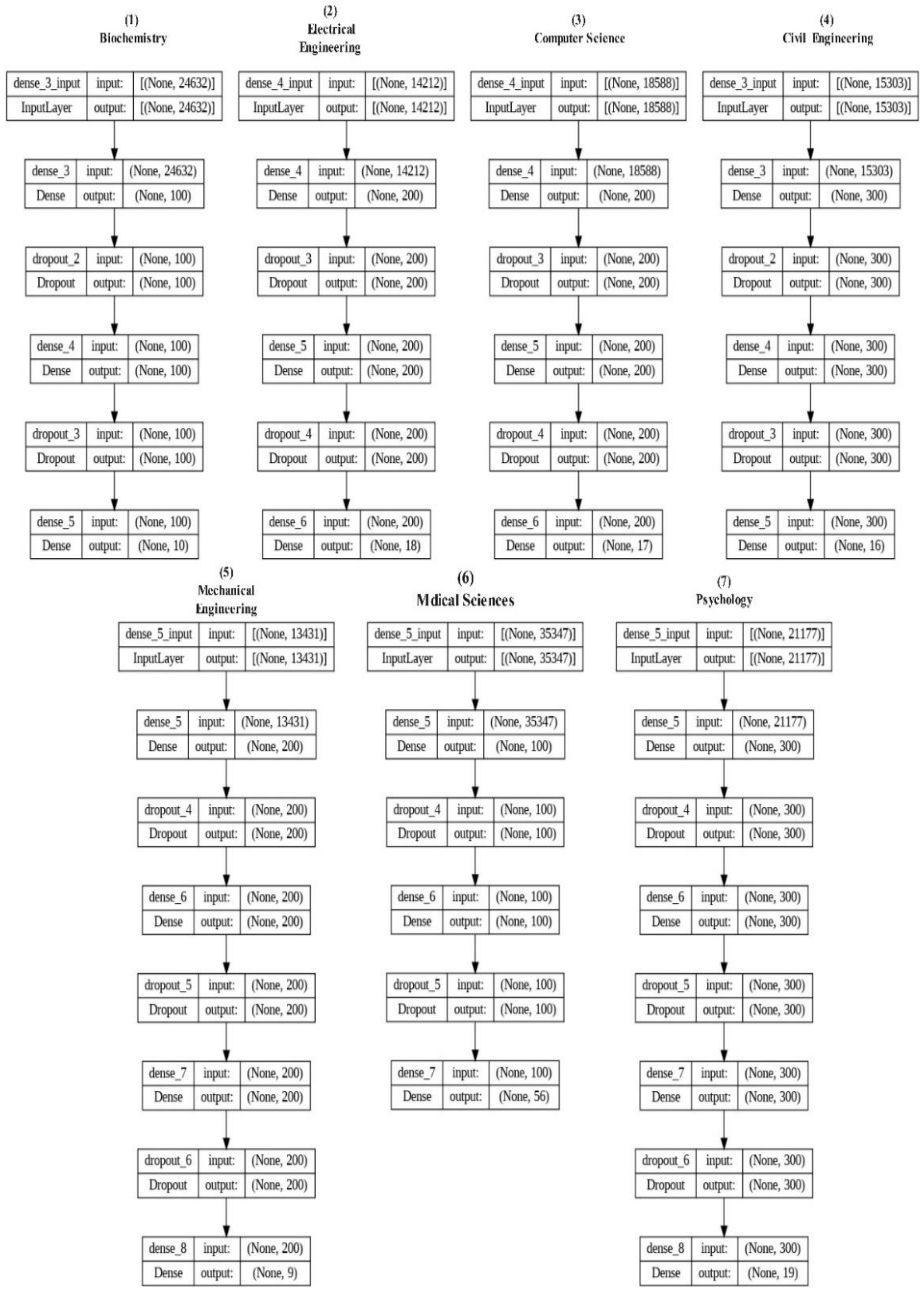

(1) Biochemistry
dense_3_input | InputLayer | input: [(None, 24632)] | output: [(None, 24632)]
dense_3 | Dense | input: (None, 24632) | output: (None, 100)
dropout_2 | Dropout | input: (None, 100) | output: (None, 100)
dense_4 | Dense | input: (None, 100) | output: (None, 100)
dropout_3 | Dropout | input: (None, 100) | output: (None, 100)
dense_5 | Dense | input: (None, 100) | output: (None, 10)
(2) Electrical Engineering
dense_4_input | InputLayer | input: [(None, 14212)] | output: [(None, 14212)]
dense_4 | Dense | input: (None, 14212) | output: (None, 200)
dropout_3 | Dropout | input: (None, 200) | output: (None, 200)
dense_5 | Dense | input: (None, 200) | output: (None, 200)
dropout_4 | Dropout | input: (None, 200) | output: (None, 200)
dense_6 | Dense | input: (None, 200) | output: (None, 18)
(3) Computer Science
dense_4_input | InputLayer | input: [(None, 18588)] | output: [(None, 18588)]
dense_4 | Dense | input: (None, 18588) | output: (None, 200)
dropout_3 | Dropout | input: (None, 200) | output: (None, 200)
dense_5 | Dense | input: (None, 200) | output: (None, 200)
dropout_4 | Dropout | input: (None, 200) | output: (None, 200)
dense_6 | Dense | input: (None, 200) | output: (None, 17)
(4) Civil Engineering
dense_3_input | InputLayer | input: [(None, 15303)] | output: [(None, 15303)]
dense_3 | Dense | input: (None, 15303) | output: (None, 300)
dropout_2 | Dropout | input: (None, 300) | output: (None, 300)
dense_4 | Dense | input: (None, 300) | output: (None, 300)
dropout_3 | Dropout | input: (None, 300) | output: (None, 300)
dense_5 | Dense | input: (None, 300) | output: (None, 16)
(5) Mechanical Engineering
dense_5_input | InputLayer | input: [(None, 13431)] | output: [(None, 13431)]
dense_5 | Dense | input: (None, 13431) | output: (None, 200)
dropout_4 | Dropout | input: (None, 200) | output: (None, 200)
dense_6 | Dense | input: (None, 200) | output: (None, 200)
dropout_5 | Dropout | input: (None, 200) | output: (None, 200)
dense_7 | Dense | input: (None, 200) | output: (None, 200)
dropout_6 | Dropout | input: (None, 200) | output: (None, 200)
dense_8 | Dense | input: (None, 200) | output: (None, 9)
(6) Mdical Sciences
dense_5_input | InputLayer | input: [(None, 35347)] | output: [(None, 35347)]
dense_5 | Dense | input: (None, 35347) | output: (None, 100)
dropout_4 | Dropout | input: (None, 100) | output: (None, 100)
dense_6 | Dense | input: (None, 100) | output: (None, 100)
dropout_5 | Dropout | input: (None, 100) | output: (None, 100)
dense_7 | Dense | input: (None, 100) | output: (None, 56)
(7) Psychology
dense_5_input | InputLayer | input: [(None, 21177)] | output: [(None, 21177)]
dense_5 | Dense | input: (None, 21177) | output: (None, 300)
dropout_4 | Dropout | input: (None, 300) | output: (None, 300)
dense_6 | Dense | input: (None, 300) | output: (None, 300)
dropout_5 | Dropout | input: (None, 300) | output: (None, 300)
dense_7 | Dense | input: (None, 300) | output: (None, 300)
dropout_6 | Dropout | input: (None, 300) | output: (None, 300)
dense_8 | Dense | input: (None, 300) | output: (None, 19)

Fig. 3 The proposed Child-level architecture of the best model

As shown in Fig. 4, the medical sciences and psychology have the highest number of records. Each domain has several areas, as shown in Table 3.

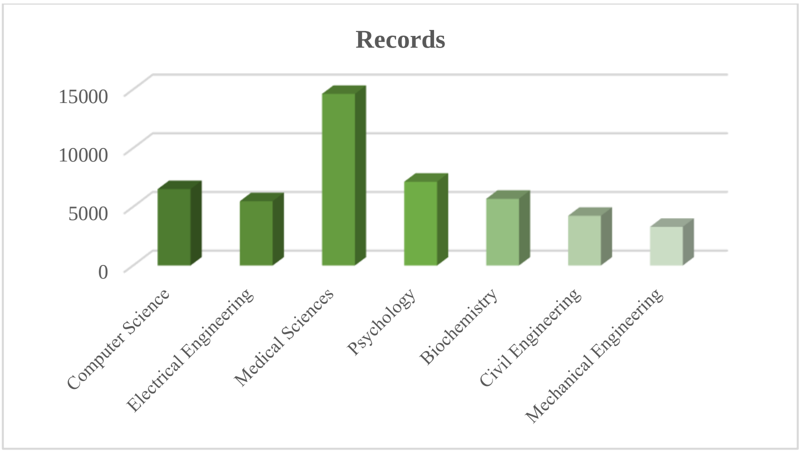


Fig. 4 Number of documents for each domain in the WOS46985 dataset [20]

Table 3 Number of areas for each domain in the WOS46985 dataset [20]

Domain	No. Areas
Computer Science	17
Electrical Engineering	18
Medical Sciences	56
Psychology	19
Biochemistry	10
Civil Engineering	16
Mechanical Engineering	9
Total	145

As shown in Fig. 5, the medical sciences and psychology have the most areas in the dataset.

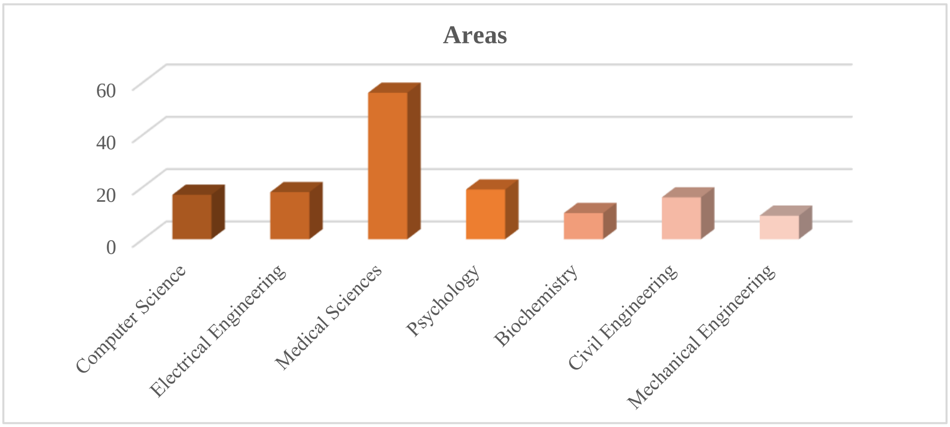


Fig. 5 Number of documents for each domain in the WOS46985 dataset [20]

3.1.2. AoI2WoS Dataset

Google Scholar Profiles (GSP) provide a very efficient way for researchers to showcase their academic achievements, publications, and research interests. It is possible to check who is citing articles, graph citations over time, and report several citation metrics that can be used in scientometrics. One important section in each GSP is called "Areas of interest". It can include a maximum of five scientific fields for everyone. Since these fields are filled manually by the authors based on their perception and knowledge about the broad scientific disciplines, they are not necessarily matched to WOS categories of sciences including 7 Domains and 134 Areas. As a result, we have gathered and created a public dataset called "AoI2WoS: Mapping Area of Interest in Google Scholar Profile to WOS categories [23]. It classifies various scientific fields extracted from 85,285 GSP records into 7 domains and 134 areas, based on WOS categories. AoI2WoS dataset spans various fields, with Biochemistry leading at 29,951 records, followed by Medical Sciences with 14,872. Computer Science contributes 10,978 records, while Civil and Mechanical Engineering offer 11,064 and 10,922 respectively as shown in Fig. 6. Psychology provides 4,305 records, and Electrical Engineering adds 3,192. Together, these domains total 85,284 records, showcasing diverse academic pursuits as shown in Table 4.

Table 4 The Number of Records for Each Domain in AoI2WoS Dataset [23]

Domain	No. Records
Computer Science	10978
Electrical Engineering	3192
Medical Sciences	14872
Psychology	4305
Biochemistry	29951
Civil Engineering	11064
Mechanical Engineering	10922
Total	85284

As shown in Fig. 6, the biochemistry has the most records. Each domain has several areas. As shown in Table 5, medical science has the most areas in the AoI2WoS dataset. In addition, Table 6 shows a bunch of samples in the AoI2WoS dataset.

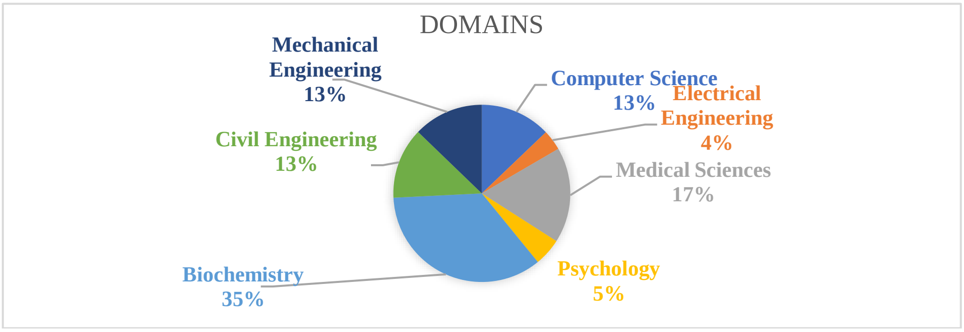


Fig. 6 The Number of Records for Each Domain in AoI2WoS Dataset [23]

Table 5 The Number of Areas for Each Domain in AoI2WoS Dataset [23]

Domain	No. Areas
Computer Science	17
Electrical Engineering	18
Medical Sciences	56
Psychology	19
Biochemistry	10
Civil Engineering	16
Mechanical Engineering	9
Total	145

Table 6 Some examples of AoI2WoS Dataset [23]

Sentence (Area of Interest in GSP)	Domain	Area
['Medical Image Analysis, 'Biomedical Image Analysis', 'Medical Image Computing', 'Image Processing']	CS	Image processing
['Power System', 'Renewable Energy', 'Natural Gas', 'Electricity Market', 'Energy Infrastructure']	ECE	Electricity
['Biostatistics', 'statistics', 'diabetes', 'cancer', 'data science']	Medical	Diabetes
['organizational behavior', 'organizational psychology', 'job design', 'creativity']	Psychology	Leadership
['Savanna Ecology', 'Grasslands', 'Grazing Ecosystems', 'Serengeti', 'Plant Ecology']	Biochemistry	Genetics
['Environmental Science', 'Chemistry', 'Soil Science', 'Water Quality']	Civil	Water Pollution
['Heat and Mass Transfer, 'Porous Media', 'Biomedical', 'Electronics Cooling', 'Heat Pipe']	MAE	Fluid mechanics

3.2. Preprocessing

In our study, we used the Natural Language Toolkit (NLTK library) to preprocess text records. According to Fig. 7, text classification includes several essential steps [24-26]. First, we used the Beautiful Soup library to remove HTML, and XML tags from the text. Then, we bumped off stop words in the 2d step. Next, we used the NLTK library to remove punctuation marks from the textual content because they do not add beneficial information to the model. Finally, we did stemming and lemmatization strategies within the closing step to reduce phrases with comparable meanings.

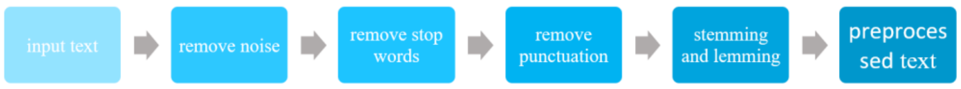


Fig. 7 Preprocessing steps in the proposed model for text classification [24,25]

3.3. Hierarchical Text Classification (HTC)

TC tasks usually involve a constrained wide variety of classes. However, TC tasks in more classes present specific challenges that include area and accuracy barriers. In the real world, many issues require multi-label classification, where Hierarchical Classification

(HC) comes into play [27]. HC entails organizing items into hierarchical levels or categories. When using TC inside an HC framework, we call it Hierarchical Text Classification (HTC). HC offers three distinct strategies: Flat approach, Local approach, and Global approach.

3.3.1. Flat approach

The flat technique is a straightforward approach to handling hierarchical classification problems. It simplifies the process by ignoring the hierarchical relationships between classes and focusing solely on predicting the most specific classes at the bottom level of the hierarchy. This method treats each class independently, without considering how classes are related within the hierarchy, as proven in Fig. 8, the flat technique only aims to predict the labels at the leaf level, disregarding the broader hierarchical context. While this approach can be useful when the hierarchical structure does not provide significant additional information, or when the primary goal is to predict specific leaf classes, it may not fully leverage the hierarchical relationships. Consequently, this can lead to suboptimal performance compared to more sophisticated hierarchical classification methods.

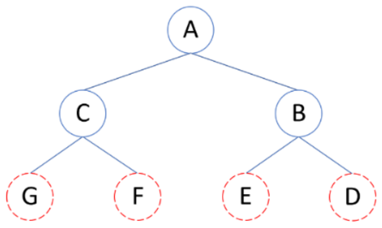


Fig. 8 Flat approach for a hierarchical classification problem

3.3.2. Local approach

The local technique stands out by leveraging the hierarchical structure to create classifiers that focus specifically on local information related to individual nodes. In this approach, each classifier is trained on data pertinent to a particular node in the hierarchy. In the context of this paper, we use the Local Classifier approach for HTC [28]. This method enables a more detailed analysis by focusing on specific nodes within the hierarchy, allowing us to handle a broader dataset more effectively, as illustrated in Fig. 9.

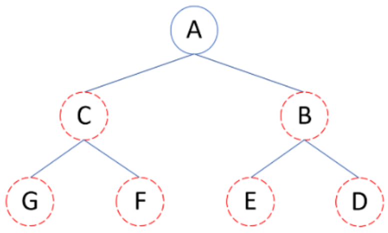


Fig. 9 Local approach for hierarchical classification problem

3.3.3. Global approach

Unlike the Local method, which focuses on specific nodes, the Global method constructs a single comprehensive model using all training data. This approach considers the entire hierarchy of classes simultaneously, rather than just one level [29]. This shows how distinctive classes are linked throughout the hierarchy, giving a better view of the facts. By thinking about the hierarchy as a whole, the global method is familiar with the larger image and the means in the back of specific training, which facilitates classifying things more as they should be. The global approach aims to improve both the accuracy and efficiency of text classification. Evaluations are conducted to compare this comprehensive approach with traditional methods to determine its strengths and potential limitations in hierarchical text classification. It is important to note that the global approach may require more computational resources and time to train, it could provide more accurate classifications and a better understanding of the data, as illustrated in Fig. 10.

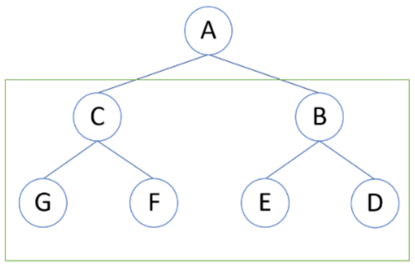


Fig. 10 Global approach for hierarchical classification Problem

3.4. Deep learning models

3.4.1. Recurrent Neural Network (RNN)

RNNs are a type of artificial neural network that is particularly well-suited for data presented in sequences [30, 31]. We use RNNs and try to optimize the RNN structure found in massive experiments. We build a hierarchy of categories, and to do that, we add a unique layer referred to as GRU (Gated Recurrent Unit) at each of the primary and secondary levels of our model. This GRU layer is a tweak on the usual RNN setup. It allows for addressing a common problem referred to as the vanishing gradient difficulty. It allows for higher training, in particular in shooting lengthy-term relationships in sequences of records, as shown in Fig. 11. By using the RNN setup along with the GRU layer, our purpose is to reinforce the overall performance and accuracy of our model based on a hierarchical shape. This approach helps us process and analyze sequences of records appropriately inside our research context, ultimately primary to superior effects and insights.

The general formulation of an RNN is expressed by Equation (1), where x_t represents the state at time t, and u_t refers to the input at step t:

$$x_t = F(x_{t-1}, u_t, \theta) \tag{1}$$

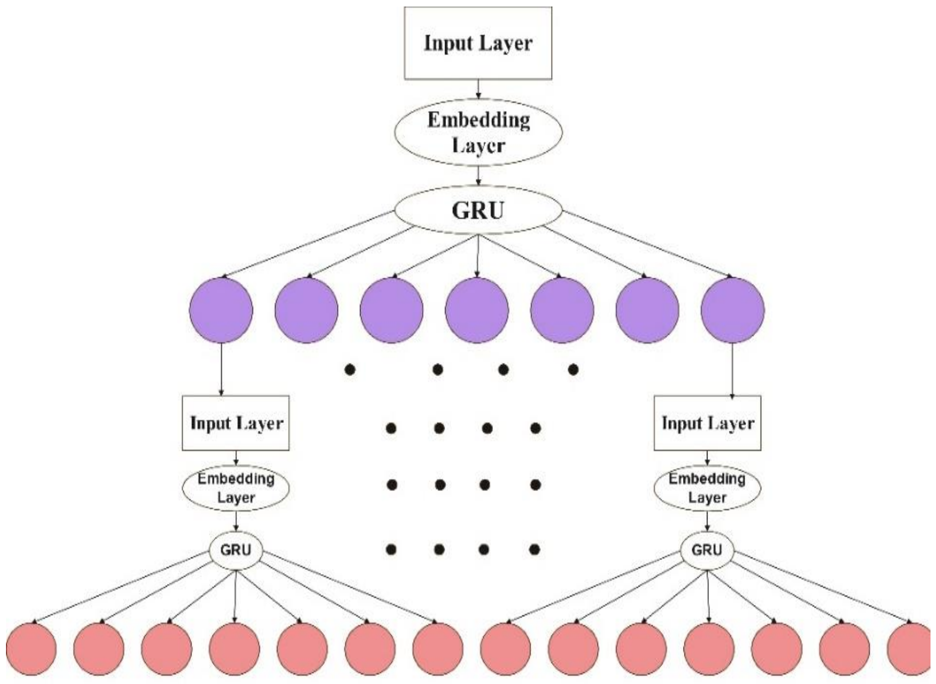


Fig. 11 The proposed RNN architecture for hierarchical text classification

3.4.2. Convolutional Neural Network (CNN)

CNNs are well-known for their effectiveness in obligations consisting of studying images, audio, speech, and [32, 33]. However, in our examination, we were focusing on how we can apply CNN structure mainly to categorize textual content. CNNs use different crucial layers such as the convolutional layer, the pooling layer, and the fully connected (FC) layer to do their task nicely [34].

Fig. 12 shows our proposed CNN architecture for hierarchical deep-learning text classification initiated with the embedding part. Then, a Conv1D layer is followed by a Maxpooling1D. This makes an efficient Embedding+ Conv1D+ Maxpooling1D combination that doubled before using the flat layer. The convolutional layer is like a detective because it reveals essential parts of the text. It does this via searching closely at the textual content with unique equipment to discover critical things, which is known as feature mapping. Then, the pooling layer makes these critical things smaller. It removes things that are not needed to foster model work better and avoid errors. Lastly, we have the wholly linked (FC) layer, which decides how all the essential parts determined using the sooner layers match together. This helps Fig. out which group the text belongs to. Besides, we have adapted our CNN setup to work well with text. Finally, it is worth mentioning that as you can see in Fig. 12, the designed structure for the parent level is followed by the child level for simplicity.

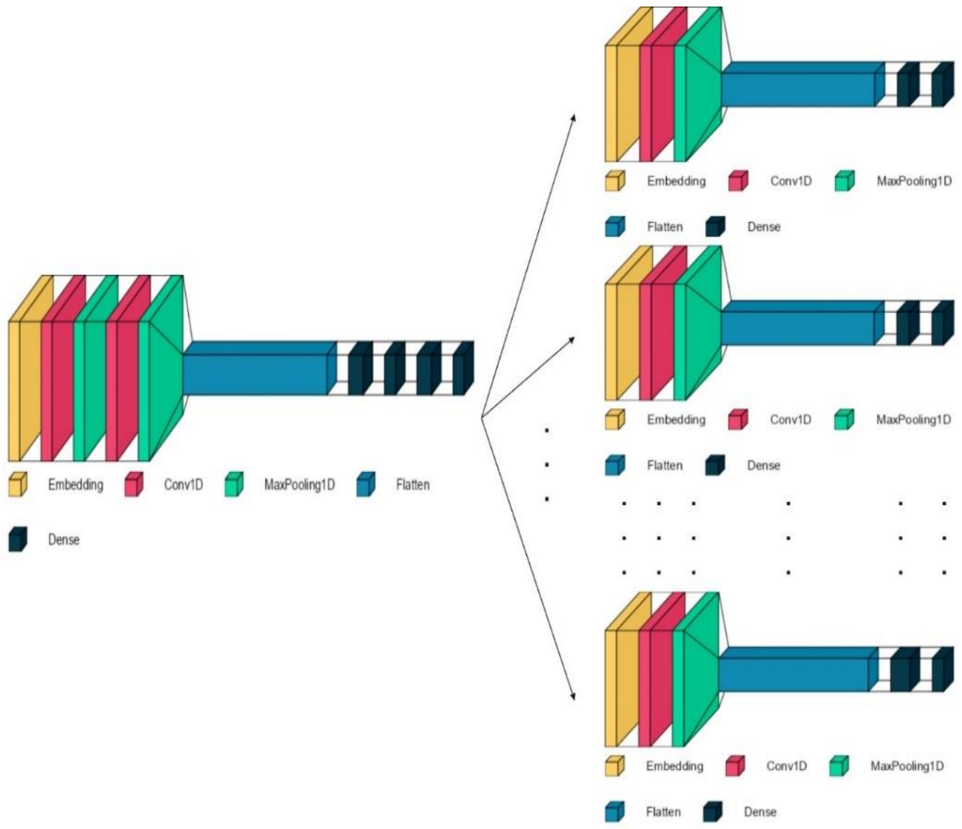


Fig. 12 The proposed CNN architecture for hierarchical text classification

3.4.3. Deep Neural Network (DNN)

In our proposed DNN setup, each layer takes data from the previous layer and passes it on to the subsequent one. We use fully connected layers, also referred to as Dense Layers, in our design. These layers are vital because they help spread data, and pick out essential functions. Also, to pick out these features, we use a method called counter-vectorization [35, 36]; this changes the text into numbers, which the DNN can understand. In our hierarchical class method, we first train a model using all the documents we have. Then, for the models in the second level of our hierarchy, we train each one with specific documents about its area. For instance, if the first level is a medical domain, the DNN model within the 2d level most effectively learns from scientific files. This targeted training ensures that every model concentrates on the place it is meant to categorize. This approach allows us to effectively utilize area-specific information, leading to strong overall performance from our hierarchical classification models in Fig. 13.

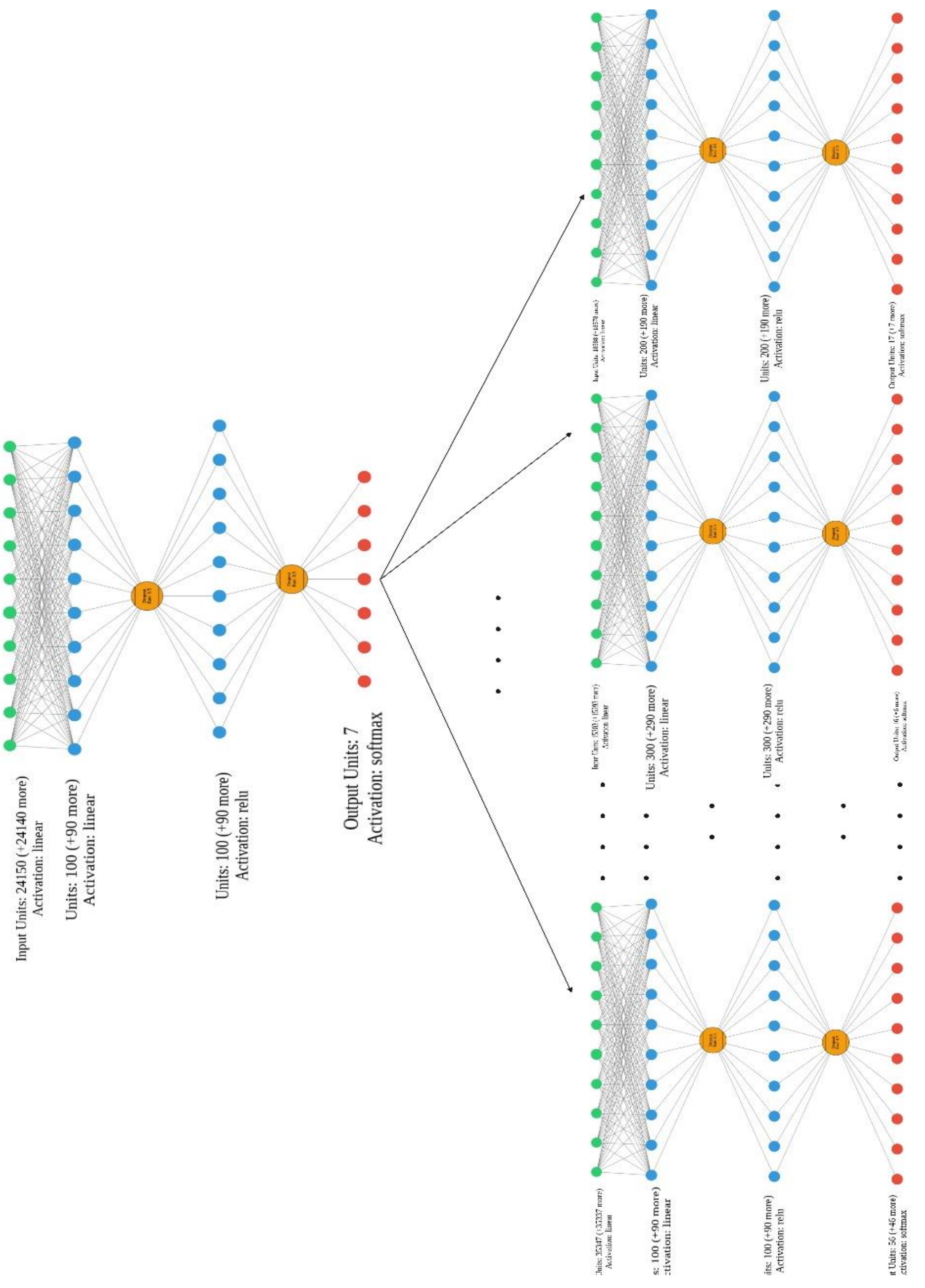


Fig. 13 The proposed DNN architecture for hierarchical text classification

Furthermore, DNNs involve the principal backpropagation algorithm. Two different activation functions, sigmoid (Equation (2)), and Rectified Linear Unit (ReLU) (Equation (3)), and we used the Softmax function for output (Equation (4)).

$$f(x) = \frac{1}{1+e^{-x}} \in (0,1) \tag{2}$$

$$f(x) = \max(0, x) \tag{3}$$

$$\sigma(z)_j = \frac{e^{Zj}}{\sum_{k=1}^{k} e^{Z_k}} \forall j \in \{1,...,k\} \tag{4}$$

3.5. Evaluation metrics

We used the following hierarchical confusion matrix [16, 37] (Equation (5) to Equation (8)) to evaluate deep learning models in hierarchical measure:

- **True positives** (TP) indicate the number of nodes correctly labeled positively by the predicted path.
- **True negatives** (TN) indicate the number of nodes correctly labeled negatively by the predicted path.
- **False positives** (FP) indicate the number of nodes incorrectly labeled positively by the predicted path.
- **False negatives** (FN) indicate the number of nodes incorrectly labeled negatively by the predicted path.

$$TP = \sum_{o_d \in \Omega} TP_H P_d, W_d \tag{5}$$

$$FP = \sum_{o_d \in \Omega} FP_H \{P_d, W_d\} 7 \tag{6}$$

$$TN = \sum_{o_d \in \Omega} TN_H P_d, W_d \tag{7}$$

$$FN = \sum_{o_d \in \Omega} FN_H \{P_d, W_d\} \tag{8}$$

Where Ω indicates the set of data, o_d indicates a single data, p_d refers to a set of all predicted allocation paths for o_d, W_d refers to the set of all true allocation paths for o_d. In the following, some significant performance metrics widely used in classification problems are formulated from Equation (9) to Equation (12):

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN} \tag{9}$$

$$Precision = \frac{TP}{TP+FP} \tag{10}$$

$$Recall = \frac{TP}{TP+FN} \tag{11}$$

$$F1-Score = \frac{2TP}{2TP+FP+FN} \tag{12}$$

Furthermore, we used macro-average to evaluate the model in flat measure [38].

4. Experimental Results

We performed the research using Google Colab, which offered a computational environment with 12 GB RAM, and 16 GB GPU. Python 3.9.6 was the chosen programming language for implementation and we used Keras Version 3, TensorFlow version 2.17.0, and the Natural Language Toolkit (NLTK) library. In addition, we used the Hyperband Tuning algorithm to choose the best hyperparameters.

For the CNN architecture, we delved into various configurations. This covered considering four candidates for the number of neurons in each dense layer, including 100, 200, 300, and 400. Additionally, we examined three candidates for the number of dense layers 1, 2, and 3 with two candidates for the wide variety of convolutional layers, and max pooling layers, every with 1, 2. Furthermore, we evaluated the filter size with two options: 3 and 5. In the case of the DNN architecture, we also investigated extraordinary setups. We tested four values for the range of neurons in every dense layer, specifically 100, 200, 300, and 400. Additionally, we explored three applicants for dense layers 1, 2, and 3. For the RNN structure, the focus changed to the number of neurons in each GRU layer. We considered three candidates, 100, 200, and 300. Furthermore, we set one candidate for the range of dense layers, which becomes 1. In words of feature extraction, we applied numerous strategies along with Glove [39], Word2Vec [40], and Fast Text [41] for the embedding layers in different architectures.

4.1. Word Embedding Dimension

Glove [42] is another broadly used technique that constructs word embeddings via leveraging global word co-occurrence statistics from a corpus. It creates word vectors by considering the co-occurrence probabilities of words inside a corpus. Achieved results in Table 7 found on the first scenario (S1 in Table 1) indicate that when the dimensions of the glove vector increase, the accuracy and disk usage of the model also expand.

Table 7 Comparison results of our RNN and DNN models in the Word Embedding Dimension due to the first scenario (S1 in Table 1)

First-level	Second-level	Word embedding	Accuracy	F1-Score	Disk-usage
RNN	DNN	glove.50d	98.46	87.28	329.59
RNN	DNN	glove.100d	98.93	91.1	353.48
RNN	DNN	glove.200d	98.9	90.8	401.26
RNN	DNN	glove.300d	99.18	93.06	449.04

As shown in Fig. 14, the glove embedding with the 300 dimensions has the best accuracy.

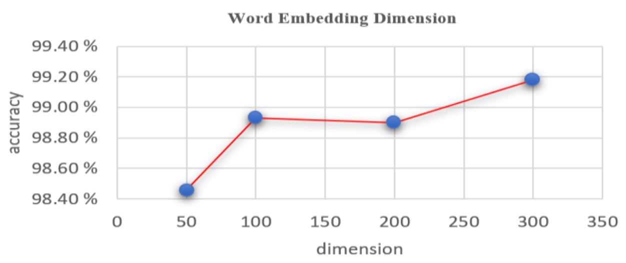


Fig. 14 Evaluation of the effect of dimension variations on the accuracy of Word Embedding in S1

However, we can see in Table 8 that when we use CNN architecture at the first level or second level, the model takes up more space than other models.

Table 8 Comparison results of proposed DNN and CNN with different Word Embedding dimensions in the first scenario (S1 in Table 1)

First-level	Second-level	Word embedding	Accuracy	F1-Score	Disk-usage
DNN	CNN	glove.50d	98.92	90.86	559.98
DNN	CNN	glove.100d	99.04	91.84	754.52
DNN	CNN	glove.200d	99.09	92.26	1074.64
DNN	CNN	glove.300d	99.1	92.4	1476.26

4.2. Word Embedding techniques

Word2Vec is a well-known approach that learns word embeddings by way of training a neural network to predict words based totally on their context (skip-gram) or to expect context words given a goal word (Continuous Bag of Words - CBOW) [43]. The word vectors create an understanding of how words are associated with each different means. Table 9 shows the outcomes of the second scenario (S2).

Table 9 Comparing different proposed models based on RNN-DNN, and the second scenario (S2 in Table 1)

First-level	Second-level	Word embedding	Accuracy	F1-Score	Disk-usage
RNN	DNN	glove.300d	99.18	93.06	449.04
RNN	DNN	Word2vec	98.81	90.3	-
RNN	DNN	Fast Text	98.53	87.97	-

It suggests that the word2vec method has a higher overall performance than fast text, but the Glove technique has an overall better performance, as shown in Fig. 15.

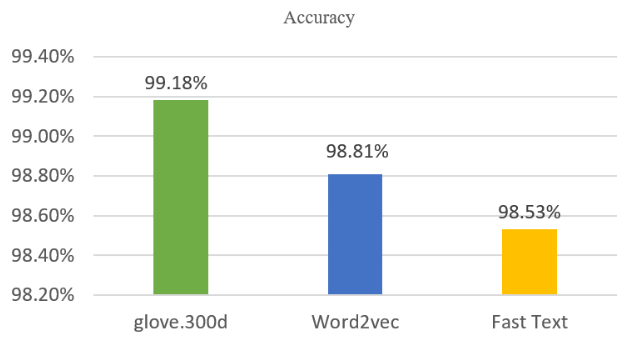


Fig. 15 Comparison results of word embedding techniques in the second scenario (S2 in Table 1).

However, we can see new results in Table 10 when we used other architecture in first level or second level.

Table 10 Comparing different techniques in other architecture according to the second scenario (S2 in Table 1)

First-level	Second-level	Word embedding	Accuracy	F1-Score	Disk-usage
DNN	CNN	glove.300d	99.1	92.4	449.04
DNN	CNN	Word2vec	99.13	92.6	-
DNN	CNN	Fast Text	99.02	91.68	-

The word2vec technique performs better than the glove technique, as shown in Fig. 16.

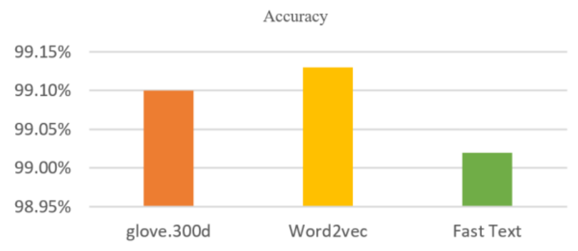


Fig. 16 Comparison results of Word embedding techniques in S2

4.3. Document Embedding Dimension (Distributed Memory)

There are two options in the Doc2Vec model. PV-DBOW (Distributed Bag of Words) disregards word order and predicts randomly sampled words from a document [44]. On the other hand, PV-DM (Distributed Memory) generates word sequences and predicts words based on the context of the passage. As shown in Table 11, the results follow a similar trend to that observed with the Glove method: as the document resolution increases, both model accuracy and disk usage increase. This trend highlights the relationship between dimensionality and accuracy in document embedding techniques, and the parallel increase in disk space usage as dimensionality grows, also seen in the Glove vocabulary.

Table 11 Increase the dimension according to the third scenario (S3 in Table 1)

First-level	Second-level	Word embedding	Accuracy	F1-Score	Disk-usage
RNN	DNN	doc2vec50	94.27	58.10	23.12
RNN	DNN	doc2vec100	94.83	62.04	36.32
RNN	DNN	doc2vec200	95.03	63.62	51.75
RNN	DNN	doc2vec300	95.31	64.86	77.63

As shown in Fig. 17, the doc2vec embedding with the 300 dimension has the best accuracy.

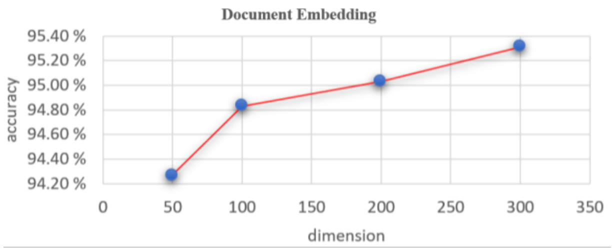
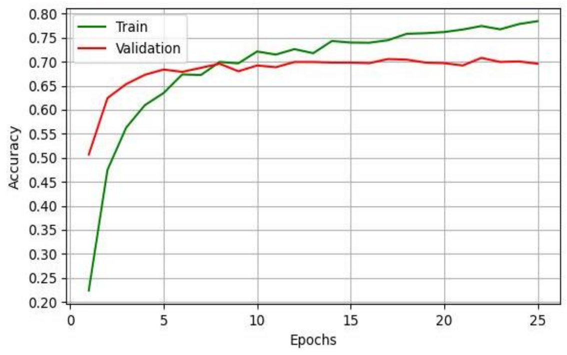
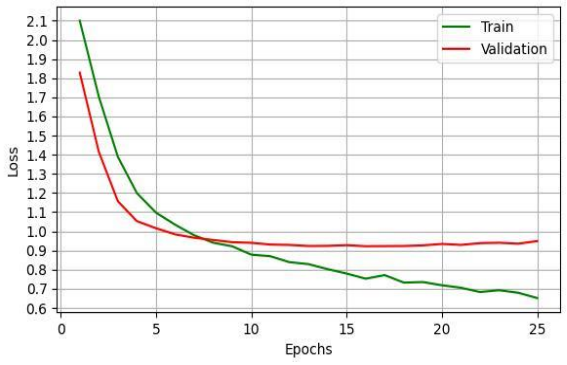


Fig. 17 Document embedding dimension in S3

In Fig. 18, we can see the accuracy and loss validation curves for the one-child architecture of the best model in this scenario, utilizing the DNN architecture. The model was trained over 25 epochs, which allows us to observe the learning process over time. As the number of epochs increases, we can see a steady improvement in accuracy, while the loss gradually decreases, indicating that the model is learning effectively. However, after a certain point, the rate of improvement slows, suggesting that the model is approaching convergence. This balance between accuracy and loss provides insight into the model's performance and its ability to generalize to unseen data.



(a)



(b)

Fig. 18 One child architecture of the best model in the Document Embedding Dimension in S3

4.4. Document Embedding Dimension (Avg, Max, Min)

Common word embeddings are extensively utilized in various NLP applications and are highly effective for generating document embeddings [45]. The overall semantic meaning of the textual content is captured by averaging the word embedding vectors within a record [46].

Table 12 provides obtained results due to S4, as the dimensions of the report embedding vector increase, both the accuracy and the disk utilization show an upward fashion. This indicates that better-dimensional embedding vectors correspond to stepped-forward accuracy, however they necessitate more disk area for the garage. This courting shows a trade-off between model overall performance and resource utilization, in which better accuracy comes at the fee of extended garage necessities.

Table 12 Increase the dimension according to the fourth scenario (S4 in Table 1)

First-level	Second-level	Word embedding	Accuracy	F1-Score	Disk-usage
RNN	DNN	Doc2Vecglove.50d	94.36	57.99	41.34
RNN	DNN	Doc2Vecglove.100d	94.69	60.23	72.52
RNN	DNN	Doc2Vecglove.200d	94.93	60.93	136.4
RNN	DNN	Doc2Vecglove.300d	95.53	64.75	190.39

According to Fig. 19, Doc2Vec embedding with a dimensionality of 300 exhibits the highest accuracy among the different embedding dimensions.

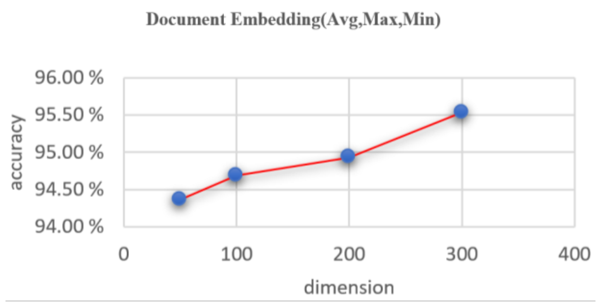


Fig. 19 Document embedding dimension in S4

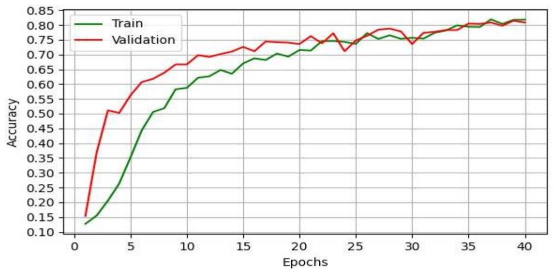
4.5. Document Embedding techniques

The Table 13 shows that the Doc2Vec (Word2vec) technique has better performance than Doc2Vec (glove)

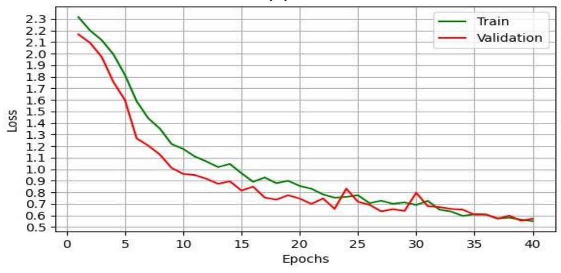
Table 13 Comparing different techniques in RNN and DNN according to the fifth scenario (S5 in Table 1)

First-level	Second-level	Word embedding	Accuracy	F1-Score	Disk-usage
RNN	DNN	Doc2Vecglove.300d	95.53408	64.75897	190.39
RNN	DNN	Doc2Vec (Word2vec)	95.54036	67.74503	199.03

Fig. 20 presents a comparison of the training and validation accuracy and loss for the one-child architecture, based on the best model identified in the fifth scenario using the DNN architecture. The model was trained for 40 epochs, during which we can observe the trends in both accuracy and loss. Initially, both metrics show significant improvements.



(a)



(b)

Fig. 20 One child architecture of the best model in Document Embedding techniques according to the fifth scenario (S5 in Table 1)

4.6. Considering all methods in the sixth scenario

When all models are compared then the best models for each scenario can be found. As shown in Table 14, the methods of document embedding are not as accurate as the methods of word embedding, but the methods they adopt so book entry are much less than word embedding methods. It can be concluded that using document embedding techniques is appropriate when space limitations are a consideration, and to achieve higher accuracy, it is advisable to use word embedding techniques (See Fig. 21).

Table 14 Result of different implementations according to the sixth scenario (S6 in Table 1)

First-level	Second-level	Word embedding	Accuracy	F1-Score	Disk-usage
RNN	DNN	doc2vec300	95.31	64.86	77.63
RNN	DNN	Doc2Vec (Word2vec)	95.5	67.74	199.03
RNN	DNN	glove.300d	99.18	93.06	449.04

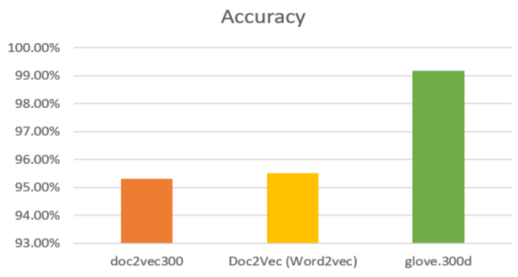
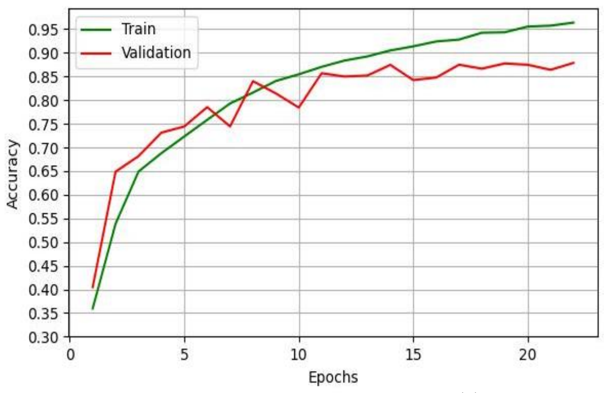
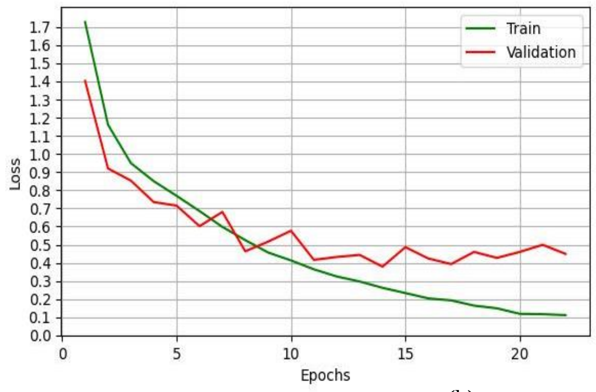


Fig. 21 Comparing the best of each scenario to the sixth scenario (S6 in Table 1)

In Fig. 22, we can see the accuracy and loss validation of the parent node of the best model in this scenario with CNN architecture in Flat Measure.



(a)



(b)

Fig. 22 Accuracy and loss validation of parent node with CNN architecture in S6

We used four epochs, and a batch size of 128 in the Hyperband Tunning algorithm to find the best architecture for each level [47]. Moreover, we used 22 epochs, and a batch size of 64 with the early stopping method. Results of train, and validation according to S6 are shown in Table 15.

Table 15 Result of train and validation in S6

Epochs	Loss		Accuracy	
	Train	Validation	Train	Validation
1	1.7472	1.1938	0.3573	0.519
2	1.168	1.0314	0.5299	0.6044
3	0.9532	0.9209	0.645	0.6426
4	0.861	0.8131	0.6831	0.6942
5	0.7876	0.7903	0.7139	0.7162
6	0.7064	0.7045	0.7486	0.75
7	0.6356	0.6545	0.7782	0.7772
8	0.5506	0.5181	0.808	0.8206
9	0.4841	0.4622	0.8322	0.841
10	0.4264	0.5237	0.8495	0.8276
11	0.3829	0.4728	0.8621	0.8364
12	0.3405	0.4126	0.8775	0.8602
13	0.3042	0.4645	0.891	0.8505
14	0.2727	0.5126	0.899	0.8312
15	0.2445	0.4002	0.912	0.8664
16	0.2193	0.4508	0.9192	0.8534
17	0.1841	0.5335	0.9301	0.829
18	0.1758	0.4381	0.9332	0.8711
19	**0.1504**	**0.4558**	**0.9436**	**0.872**
20	0.151	0.4872	0.9416	0.8585
21	0.1238	0.5729	0.9526	0.8374
22	0.1151	0.5194	0.9611	0.8634

Table 16 compares flat measures in S6 across distinctive model configurations. Our proposed model that utilizes RNN for both the first, and second levels with word2vec, reaches an accuracy of 87.62% for parents, and 85.02% for models at the child level. Moreover, the F1-Score metrics are reported at 87.63% for the parent-level, and 84.95% for the child level. In comparison, [20] showcases an accuracy of 90.45% at the parent level, and 84.66% at the child level.

Table 16 Comparing results in flat measurement using the WOS46985 dataset

Model	Flat Measure			
	Parent-level accuracy	Child-level accuracy	Parent-level F1-Score	Child-level F1-Score
Proposed (RNN, RNN)	87.62	85.02	87.63	84.95
[20]	90.45	84.66	-	-

Table 17 illustrates a comparison of hierarchical measures to the sixth scenario (S6), which specializes in different model configurations, and their overall performance metrics. The proposed model configurations consist of CNN on the parent, and DNN at the child level with word2vec embedding. This setup achieves an impressive accuracy of 99.33%, and an F1-Score of 94.29%. In contrast, [20] uses RNN architecture at both the parent and child levels with the glove.100d embeddings achieves lower accuracy of 98.48% %, and F1-Score of 87.54%. These results show how important the model design and word embedding are for learning tasks in this particular situation in hierarchical classification, the architecture (RNN, RNN) works better than other architectures in flat comparison but when we use a hierarchical confusion matrix, we notice that (RNN, RNN) is not the best architecture. The (CNN, DNN) in WOS46985 is better.

Table 17 Comparing results in hierarchical measurement using the WOS46985 dataset

Model	Hierarchical Measure	
	Accuracy	F1-Score
Proposed (CNN, DNN)	99.33	94.29
[20]	98.48	87.54

4.7. Achieved results based on AoI2WoS Dataset

In this section, we evaluate our model based on our developed public dataset named AoI2WoS [23]. Table 18 presents the performance metrics of an RNN classifier that utilizes Google News vectors with an embedding dimension size of 300 at the parent level. The classifier demonstrates high effectiveness in processing and categorizing data, as indicated by its training and testing metrics. During training, the model achieved a remarkable 96% accuracy, with precision and recall rates also at 96% and 95%, respectively. These metrics suggest that the model is highly reliable in predicting and classifying within the training dataset. When evaluated on the test dataset, the RNN maintained strong performance, achieving a 92% accuracy rate, along with 92% precision and 91% recall. We didn't use CNN architecture for the parent level because the length of the sentences was short.

Table 18 The Flat measure performance of parent level using AoI2WoS dataset for RNN classifier and Google News-vectors embedding.

Dimension Size	Train Accuracy	Train Precision	Train Recall	Test Accuracy	Test Precision	Test Recall
300	96%	96%	95%	92%	92%	91%

Table 19 outlines the performance metrics of a DNN classifier used at the child level of hierarchical classification models. This classifier leverages a Count Vectorizer with a dimension size of 2,439 to represent the data.

Table 19 The Flat measure performance of child level using AoI2WoS dataset for DNN classifier and Count Vectorize embedding.

Dimension Size	Train Accuracy	Train Precision	Train Recall	Test Accuracy	Test Precision	Test Recall
2439	86%	92%	81%	83%	88%	78%

The hierarchical confusion matrix was employed to evaluate the entire model, providing a detailed assessment of its performance across all classification levels as shown in Table 20.

Table 20 Comparing Results in hierarchical measurement using AoI2WoS dataset using Google News-vectors embedding

First-level	Second-level	TP	TN	FP	FN	Accuracy	F1-Score
RNN	DNN	156701	1963851	13867	13867	98.709	91.8701
RNN	RNN	152002	1959150	18566	18566	98.2716	89.1152

Finally, in Table 21, we compare the result of our model based on the AoI2WoS dataset with the available study in [20].

Table 21 Comparing results in hierarchical measurement using the AoI2WoS dataset

Model	Hierarchical Measure	
	Accuracy	F1-Score
Proposed (RNN, DNN)	98.70	91.87
[20]	98.48	87.54

5. Conclusion and Future Work

To solve a complex problem of hierarchical text classification in WOS scientific fields, we aimed to identify the best model based on conducting diverse scenarios and designing many deep learning architectures including CNN, DNN, and RNN. Each deep learning architecture was optimized using the Hyperband Tuning technique. Several strategies were used for the embedding layers in various designs, including Glove, Word2Vec, and Fast Text. Notably, both disk usage and model accuracy have been considerably impacted by the word embedding measurement selection. The Glove embedding with three hundred dimensions, for instance, showed high-quality accuracy for RNN, and DNN, highlighting the importance of choosing the proper dimensions. Furthermore, an intensive evaluation of word embedding techniques discovered that Word2Vec outperformed Fast Text, with Glove continuously outperforming the two. To summarize, the proposed CNN-DNN model outperforms the [20]) with an accuracy of 99.33% and an F1-score of 94.29%, compared to 98.48% accuracy and 87.54% F1-score for WOS46985 dataset. The proposed model's higher accuracy and significantly improved F1-score demonstrate its effectiveness in balancing precision and recall, making it better suited for handling class imbalances in hierarchical classification scenarios. Besides, using the AoI2WoS dataset for hierarchical measurement tasks shown that the proposed RNN-DNN model achieves a higher accuracy (98.70%) and F1-score (91.87%) compared to [20] with 98.48% accuracy and 87.54% F1-score. While the accuracy difference is small, the proposed model’s significantly higher F1-score indicates better precision and recall balance, making it more suitable for handling class imbalances or minimizing false positives and false negatives in hierarchical classification tasks. In both datasets, the architecture (RNN, RNN) works better than other architectures in classification when evaluated by flat comparison. But when we use a hierarchical confusion matrix, we notice that (RNN, RNN) is not the best architecture. The (CNN, DNN) in WOS46985 and (RNN, DNN) in the AoI2WoS dataset are more efficient and better than (RNN, RNN) architecture for hierarchical classification.

Future studies should check out more complicated deep learning embeddings, and structures, investigating their scalability to larger datasets, and a broader range of scientific fields. Potential directions for improvement include ensemble tactics for model mixing, and dynamic embeddings that modify to changing linguistic subtleties. By integrating domain-specific knowledge, promoting multidisciplinary cooperation, improving semantic textual content similarity, and knowledge hierarchical structural evolution in a systematic area [48, 49], hierarchical classification models can be progressed extra and made applicable in plenty of scientific settings. As a result, our work establishes the foundation for persistent developments in hierarchical text categories and affords a research schedule for destiny studies so one can tackle converting troubles in the area.

REFERENCES

[1] S. Hasani, M. Bahaghighat and M. Mirfatahia, "The mediating effect of the brand on the relationship between social network marketing and consumer behavior", *Acta Technica Napocensis,* vol. 60, no. 2, pp. 1-6, 2019.

[2] E. Amouee, M. Mohammadi Zanjireh, M. Bahaghighat, and M. Ghorbani, "A new anomalous text detection approach using unsupervised methods", *FU Elec. Energ.,* vol. 33, pp. 631-653, 2020.

[3] M. Ghorbani, M. Bahaghighat, Q. Xin and F. Özen, "ConvLSTMConv network: a deep learning approach for sentiment analysis in cloud computing," *J. Cloud Comput.,* vol. 9, no. 1, p. 16, 2020.

[4] A. Shamseen, M. Mohammadi Zanjireh, M. Bahaghighat and Q. Xin, "Developing a parallel classifier for mining in big data sets", *IIUM Eng. J.,* vol. 22, no. 2, pp. 119-134, 07/04 2021.

[5] M. T. Sereshki, M. M. Zanjireh and M. Bahaghighat, "Textual outlier detection with an unsupervised method using text similarity and density peak", *Acta Universitatis Sapientiae, Informatica,* vol. 15, no. 1, pp. 91-110, 2023.

[6] N. Hussain, H. Turab Mirza, G. Rasool, I. Hussain and M. Kaleem, "Spam review detection techniques: A systematic literature review", *Appl. Sci.*, vol. 9, no. 5, p. 987, 2019.

[7] R. A. Stein, P. A. Jaques and J. F. Valiati, "An analysis of hierarchical text classification using word embeddings", *Inform. Sci.,* vol. 471, pp. 216-232, 2019.

[8] F. Zhao, Z. Wu, L. He and X.-Y. Dai, "Label-correction capsule network for hierarchical text classification", *IEEE/ACM Trans. Audio Speech Lang. Proc.,* vol. 31, pp. 2158-2168, 2023.

[9] O. Birgin and K. Özkan, "Comparing the concept images and hierarchical classification skills of students at different educational levels regarding parallelograms: a cross-sectional study", *Int. J. Math. Educ. Sci. Technol.,* vol. 55, no. 4, pp. 850-882, 2024.

[10] A. Fernández, S. del Río, A. Bawakid and F. Herrera, "Fuzzy rule based classification systems for big data with MapReduce: granularity analysis", *Adv. Data Anal. Classif.,* vol. 11, pp. 711-730, 2017.

[11] A. Aldoseri, K. Al-Khalifa and A. Hamouda, "A roadmap for integrating automation with process optimization for AI-powered digital transformation", *Preprints 2023*, p. 2023101055 2023.

[12] M. Kappi and B. Mallikarjuna, "Artificial intelligence and machine learning for disaster prediction: a scientometric analysis of highly cited papers", *Natural Hazards,* pp. 1-21, 2024.

[13] F. Gargiulo, S. Silvestri, M. Ciampi and G. De Pietro, "Deep neural network for hierarchical extreme multi-label text classification," *Appl. Soft. Comput.,* vol. 79, pp. 125-138, 2019.

[14] Z. Rahimi and M. M. Homayounpour, "Tens-embedding: a tensor-based document embedding method," *Expert Syst. Appl.,* vol. 162, p. 113770, 2020.

[15] R. Miikkulainen et al., "Evolving deep neural networks," in *Artificial intelligence in the age of neural networks and brain computing*: Elsevier, 2024, pp. 269-287.

[16] P. Cavalin and L. Oliveira, "Confusion matrix-based building of hierarchical classification", in Proceedings of the *Progress in Pattern Recognition, Image Analysis, Computer Vision, and Applications: 23rd Iberoamerican Congress, CIARP 2018, Madrid, Spain, November 19-22, 2018,* pp. 271-278.

[17] L. Li, K. G. Jamieson, G. DeSalvo, A. Rostamizadeh and A. Talwalkar, "Hyperband: Bandit-Based Configuration Evaluation for Hyperparameter Optimization," in Proceedings of the *ICLR (Poster)*, 2017, p. 53.

[18] S. Minaee, N. Kalchbrenner, E. Cambria, N. Nikzad, M. Chenaghlu and J. Gao, "Deep learning--based text classification: a comprehensive review", *ACM Comput. Surveys (CSUR),* vol. 54, no. 3, pp. 1-40, 2021.

[19] J. Zhang, Y. Li, F. Shen, Y. He, H. Tan and Y. He, "Hierarchical text classification with multi-label contrastive learning and KNN", *Neurocomputing,* vol. 577, p. 127323, 2024.

[20] K. Kowsari, D. E. Brown, M. Heidarysafa, K. J. Meimandi, M. S. Gerber and L. E. Barnes, "HDLTex: Hierarchical Deep Learning for Text Classification", in Proceedings of the 16th IEEE International Conference on Machine Learning and Applications (ICMLA), 2017, pp. 364-371.

[21] N. Jindal and B. Liu, "Review spam detection," in Proceedings of the 16th International Conference on World Wide Web, 2007, pp. 1189-1190.

[22] N. C. Dang, M. N. Moreno-García and F. De la Prieta, "Sentiment analysis based on deep learning: A comparative study," *Electronics*, vol. 9, no. 3, p. 483, 2020.

[23] M. Bahaghighat and P. Jahani Rad, "AoI2WoS: Mapping Area of Interest in Google Scholar Profile to Web Of Science (WoS) Scientific Fields Categories", *Mendeley Data*, vol. V1, 2024.

[24] K. Srinidhi, A. Harika, S. S. Voodarla and J. Abhiram, "Sentiment Analysis of Social Media Comments using Natural Language Procesing", in Proceedings of International Conference on Inventive Computation Technologies (ICICT), 2024, pp. 762-768.

[25] L. Dai, J. Mao, L. Xu, X. Fan and X. Zhou, "SecNLP: An NLP classification model watermarking framework based on multi-task learning", *Comput. Speech Lang.*, vol. 86, p. 101606, 2024.

[26] Q. Li, S. Jing and L. Chen, "Hierarchical Knowledge Distillation on Text Graph for Data-limited Attribute Inference", *arXiv preprint*, no: 2401.06802, 2024.

[27] C. Montenegro, R. Santana and J. A. Lozano, "Introducing multi-dimensional hierarchical classification: Characterization, solving strategies and performance measures," *Neurocomputing*, vol. 533, pp. 141-160, 2023.

[28] R. Cerri, R. C. Barros and A. C. de Carvalho, "Hierarchical multi-label classification for protein function prediction: A local approach based on neural networks," in Proceedings of the 11th IEEE International Conference on Intelligent Systems Design and Applications, pp. 337-343.

[29] J. N. Hernandez, L. E. Sucar and E. F. Morales, "A hybrid global-local approach for hierarchical classification," in Proceedings of the 26th Florida Artificial Intelligence Research Society Conference (FLAIRS), 2013, pp. 432-437.

[30] A. Rahman, V. Srikumar and A. D. Smith, "Predicting electricity consumption for commercial and residential buildings using deep recurrent neural networks", *Appl. Energy*, vol. 212, pp. 372-385, 2018.

[31] M. Zulqarnain, R. Ghazali, M. G. Ghouse and M. F. Mushtaq, "Efficient processing of GRU based on word embedding for text classification" *JOIV: Int. J. Inform. Vis.*, vol. 3, no. 4, pp. 377-383, 2019.

[32] S. Wang, M. Huang and Z. Deng, "Densely connected CNN with multi-scale feature attention for text classification", in Proceedings of the 27th International Joint Conference on Artificial Intelligence (IJCAI), 2018, vol. 18, pp. 4468-4474.

[33] H. Wang, J. He, X. Zhang and S. Liu, "A short text classification method based on N-gram and CNN", *Chin. J. Electron.*, vol. 29, no. 2, pp. 248-254, 2020.

[34] R. Romero, P. Celard, J. M. Sorribes-Fdez, A. Seara Vieira, E. L. Iglesias and L. Borrajo, "MobyDeep: A lightweight CNN architecture to configure models for text classification," *Knowledge-Based Syst.*, vol. 257, p. 109914, 2022.

[35] I. Goodfellow, Y. Bengio and A. Courville, *Deep learning*. MIT press, 2016.

[36] J. Huang, J. Zheng, S. Gao, W. Liu and J. Lin, "Grid text classification method based on DNN neural network," in Proceedings of the International Conference on Computer Science Communication and Network Security (CSCNS2019), MATEC Web of Conferences, 2020, vol. 309, p. 03016.

[37] A. Kosmopoulos, I. Partalas, E. Gaussier, G. Paliouras and I. Androutsopoulos, "Evaluation measures for hierarchical classification: a unified view and novel approaches", *Data Min. Knowl. Discov.*, vol. 29, pp. 820-865, 2015.

[38] F. Brucker, F. Benites and E. Sapozhnikova, "An empirical comparison of flat and hierarchical performance measures for multi-label classification with hierarchy extraction", in Proceedings of the Knowledge-Based and Intelligent Information and Engineering Systems: 15th International Conference, KES 2011, Kaiserslautern, Germany, September 12-14, 2011, Proceedings, Part I 15, Springer, 2011, pp. 579-589.

[39] E. Hindocha, V. Yazhiny, A. Arunkumar and P. Boobalan, "Short-text Semantic Similarity using GloVe word embedding", *Int. Res. J. Eng. Technol.*, vol. 6, no. 4, 2019.

[40] D. Karani, "Introduction to word embedding and word2vec", *Towards Data Science*, vol. 1, 2018.

[41] I. N. Khasanah, "Sentiment Classification Using fastText Embedding and Deep Learning Model", *Procedia Comput. Sci.*, vol. 189, pp. 343-350, 2021.

[42] C. N. Mohammed and A. M. Ahmed, "A semantic-based model with a hybrid feature engineering process for accurate spam detection", *J. Electr. Syst. Inform. Technol.*, vol. 11, no. 1, p. 26, 2024.

[43] M. Khuntia and D. Gupta, "Indian News Headlines Classification using Word Embedding Techniques and LSTM Model", *Procedia Comput. Sci.*, vol. 218, pp. 899-907, 2023.

[44] J. Wang, W. Feng, Y. Chen, H. Yu, M. Huang and P. S. Yu, "Visual domain adaptation with manifold embedded distribution alignment", In Proceedings of the 26th ACM International Conference on Multimedia, 2018, pp. 402-410.

[45] Q. Le and T. Mikolov, "Distributed representations of sentences and documents," in Proceedings of the 31st International Conference on Machine Learning, 2014, pp. 1188-1196.

[46] G. Balikas and M.-R. Amini, "An empirical study on large scale text classification with skip-gram embeddings," *arXiv e-prints*, p. 1606.06623, 2016.

[47] M. Khodak, T. Li, L. Li, M. Balcan, V. Smith and A. Talwalkar, "Weight sharing for hyperparameter optimization in federated learning", In Proceedings of the International Workshop on Federated Learning for User Privacy and Data Confidentiality in Conjunction with ICML, 2020, vol. 2020.

[48] Y. Qian, Y. Liu and Q. Z. Sheng, "Understanding hierarchical structural evolution in a scientific discipline: A case study of artificial intelligence", *J. Informetrics*, vol. 14, no. 3, p. 101047, 2020.

[49] H. Teng, N. Wang, H. Zhao, Y. Hu and H. Jin, "Enhancing semantic text similarity with functional semantic knowledge (FOP) in patents", *J. Informetrics*, vol. 18, no. 1, p. 101467, 2024.