Enhancing Sequential Sentence Classification of Biomedical Abstracts Using Multilevel Embedding Approach Based on Deep Learning Methods

[1]Avijit Karn; [2]Anand Kumar Sah

[1]Department of Computer Engineering Khwopa Engineering College Libali, Bhaktapur, Nepal

[2]Department of Electronics and Computer Engineering Institute of Engineering, Pulchowk Campus, Kathmandu, Nepal

Email: karn.avijit@khec.edu.np ; anand.sah@pcampus.edu.np

(Received: September 10, 2023, Revised: December 1, 2024)

Abstract

Sequential sentence classification in natural language processing tasks has witnessed significant advancements with the integration of deep learning and transfer learning techniques. In this research, we study a deep learning model and investigate different components of it to study details on word-embedding adjustments, complexity challenges and also misclassifications in abstract sectioning. We also propose a multilevel embedding approach with some modifications done in the model that addresses the contribution of additional layers present in the studied model. Multiple deep learning architectures based on character, token and positional level embeddings are used to learn and obtain the final classification of sentences for a particular context in an abstract sequentially. These models are trained on large-scale biomedical Randomized Controlled clinical Trials (RCT) datasets to learn representations that capture domain-specific features. On validating and testing with the PubMed 20k dataset for all deep models, the Tribrid model performs significantly good with accuracy and precision of 0.856 and 0.854 respectively, which is also marginally good when compared to other baseline models. Experimental results demonstrate that the integration of multiple levels of embeddings leads to improved sequential classification performance as compared to using either embedding approach individually

Keywords: *Abstract Sectioning, Convolutional Neural Networks, Deep Learning, Multilevel Embeddings, Pretrained Models, Transfer Learning*

Introduction

The number of scientific publications being published every year has crossed above 30,000 to 2 million articles, and a total of above 50 million articles have been published till now, with half over 30 million biomedical papers being indexed through PubMed. As there is an utmost significance of these publications for providing abundant information to the researchers for finding new ways and directions in their work, along with guidance of research navigations, and also providing useful evidences for some remarkable facts, however it has also become quite tedious to take the proper advantage of all the information due to its excessiveness.

In case of retrieving or searching for some works published in previous articles, a researcher generally looks for the abstract and try to analyze it in order to gather some information according to their interest or requirements. Basically, this method of information retrieval is only most helpful when the abstracts are well structured and grouped, for instance, the textual display is well divided into meaningful headings as Background, Objectives, Method, Result and Conclusion.

In this work, we study the state-of-art deep learning model according to Jin & Szolovits (2018). The multilayered model has several layers built in an attempt to capture different aspects for the sequential sentence classification. On observing the significance of each and every layer used in this model, we will try to integrate some specific details on the methods employed and present a more comprehensive work on how some adjustments in the model could enhance its performance and provide better understanding of the approach.

We also propose the multilevel embedding approach for the sequential classification of sentences under semantic headings in medical paper abstracts. For this, use of multiple deep learning models to learn character level embeddings, token level embeddings and positional embeddings will be done, as well as combination of these three levels to obtain the final classified sequential sentences in biomedical abstract. We will be training, evaluating and testing the effectiveness of these representations based on the comparison and performance analysis of each deep model.
For this, the main work will be the training and evaluation of the multiple deep learning models using the datasets available from the PubMed 200k RCT, which is the source of data for sentence classification in medical paper's abstract.

2. Literature Review

Current approaches in natural language processing (NLP) based on artificial neural networks (ANN) eliminate the need for manual feature engineering, as they are trained automatically to learn features from both word and character embeddings (li et al., 2022). However, the majority of current studies employing ANNs for short-text classification overlook the incorporation of context. This stands in contrast to sequential sentence classification, where each sentence's classification within a text considers its contextual surroundings. The context utilized for classification may involve neighboring sentences or even the entire text. Furthermore, research on the sequential classification of sentences in medical abstracts involves examining the contribution of individual components and layers within complex models, as proposed by Jin & Szolovits (2018) This investigation extends to datasets beyond those annotated in previous studies.

In recent times, multi-level embeddings Firdaus et al. (2021) that merge representations from various units have emerged as an alternative to single-level embeddings. They aim to capture the internal structure of words, such as morphology, and facilitate robust generalization over out-of-vocabulary words. However, these representations lack the incorporation of positional embeddings, which are essential for achieving more effective sequential classification of sentences based on labels or semantic headings.

Additionally, recent research has focused on segmenting biomedical abstracts by incorporating attention mechanisms using the PubMed 200K RCT dataset Jin & Szolovits (2018). The model is quite intricate, featuring multiple layers aimed at capturing various aspects of the problem. However, there is a lack of specific details regarding the methods used for word embeddings, addressing misclassification issues, and discussing practical implications. The model heavily relies on word embeddings, and the inclusion of extra layers (attention and CRF) in sentence encoding and the top layer, respectively, as well as the absence of direct training on the character and token levels, adversely impact the accuracy of the model

3. Methodology

I. Architecture of the Multilayered Model for Sequential Sentence Classification

The multilayered model briefly mentions adjustment of word embeddings to improve the model's performance but lacks specific details on the methods employed. Providing a more comprehensive explanation of how the adjustments are made and the impact they have on the model's performance would enhance the reproducibility and understanding of the approach. Moreover, there is the limited analysis on the misclassifications made by model, analyzing the types of errors and potential causes could offer valuable insights into areas for model improvement and help guide future research directions.

In addition, the presence of attention mechanisms and CRF layers seems to only bring the extra complexity in the sentence embedding due to the addition of additional matrices and also unpredictability in this model. The model does not clearly state the evaluation metrics used to assess the performance of the model.

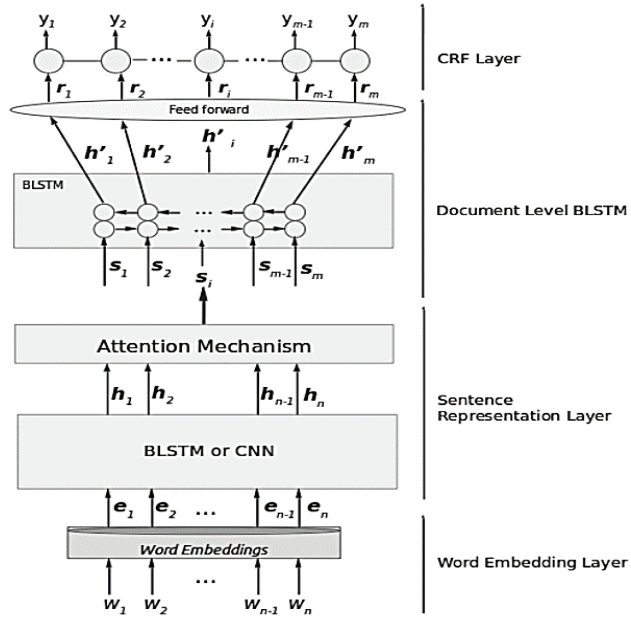


Figure 1: Multilayered Model for Sequential Sentence Classification Jin & Szolovits (2018)

Layer of Word Embeddings – For the specified word, this embedding layer corresponds to the process of associating each word with a vector in a multi-dimensional word embedding space. In this particular model, word embeddings have been pre-trained employing the Word2Vec model. In our proposed approach, we conduct embeddings for both tokens and characters utilizing a 1D CNN, culminating in the output of label probabilities (Jang et al., 2023).

Layer of Sentence Representation - This layer accepts a sequence of embedding vectors that represent a sequence of words within a sentence and forwards them to the encoder network. As mentioned, either CNN or BiLSTM networks can be utilized as the sentence encoder. In our model, we opt for BiLSTM, a component that yields cutting-edge outcomes. BiLSTM employs two sets of neurons working in opposite time directions, extracting insights from past and future hidden states. At each hidden unit corresponding to word Wi, these hidden states are combined. Given that information is gathered from both preceding and succeeding words at every position, BiLSTM is capable of grasping intricate lexical and semantic connections among words.

Document level BiLSTM and CRF - This layer accepts a sequence of sentence encoders, aligned with the sentences within a specific abstract, and produces a sequence of probability vectors. The configuration of the BiLSTM structure mirrors that of the preceding layer. Every output from the hidden units is fed into a feed-forward neural network that includes a single hidden layer. Subsequently, this network generates the probability vector. Each element k within this vector signifies the probability associated with the given sentence belonging to the Kth category. Employing a linear statistical model at the uppermost layer serves the purpose of capturing underlying patterns within the logical composition of the abstract. This arrangement proves especially valuable when focusing on recognizing sequential attributes that facilitate sentence differentiation. In this stage, the input comprises a sequence of probability vectors sourced from the preceding layer, corresponding to the sentences within a provided abstract. The outcome is a sequence of labels. The CRF (Conditional Random Field) is harnessed primarily to model the relationships existing between consecutive sentence categories.

Attention and CRF Layer contribution – The model also introduces an attention-based mechanism, presenting added complexities to the sentence embedding layer. This is attributed to the need to train two additional matrices and generate diverse sentence encoders for individual sentences. Alternatively, we employ a hybrid token embedding layer, a fusion of token embeddings and character embeddings. This construction involves creating a stacked embedding that captures sequences prior to their passage to the label layer for sequence label prediction (Tirotta et al., 2023). The CRF comprehends transition probabilities between consecutive categories. Yet, due to its position as the top layer, gradients experience alterations that propagate across preceding layers during back-propagation. This introduces a level of unpredictability into the model's behavior.

Proposed Architecture of Multilevel Embedding model for Sequential Sentence Classification

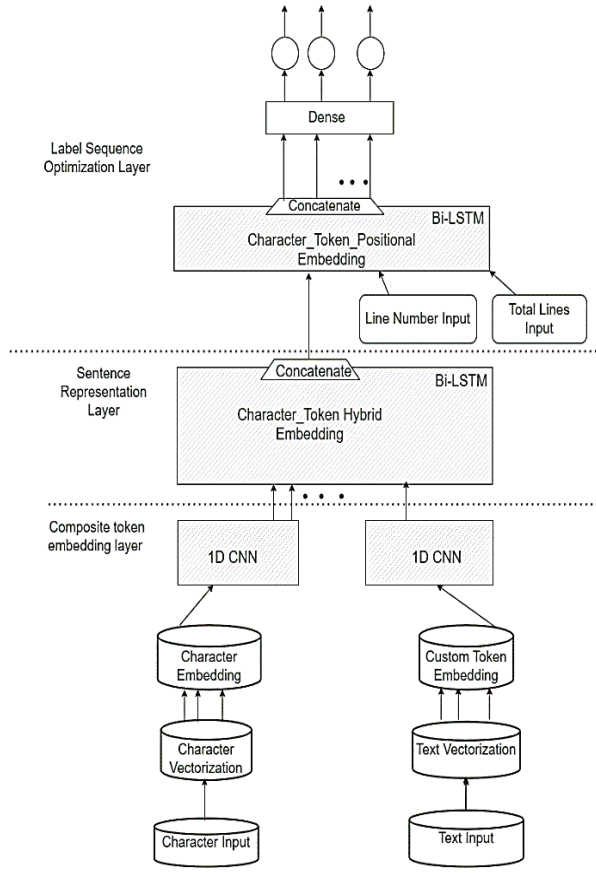


Figure 2: Proposed Multilevel Embedding Model for Sequential Sentence Classification

Deep Learning Models

Model 1: Convolution 1D model with token embeddings

Model 2: Convolution 1D model with character embeddings (Conv1D, GlobalMaxPool1D)

Model 3: Hybrid Model (Hybrid Embedding Layer), pretrained token embedding and character sequences embedding. Build using bidirectional LSTM and use of Concatenate from TensorFlow layers.

Model 4: Tribrid Model; pretrained token embeddings, character embeddings and positional embeddings.

II. Dataset Preparation

The model evaluation for the sentence classification task is done using the below medical abstract datasets from PubMed, where each sentence of the abstract is signified with one

label. We used this collection consisting of randomized controlled trials (RCTs) based medical abstracts for training, validating and testing our both baseline and deep learning models for sequential sentence classification.

PubMed 200k RCT Dataset

Dernoncourt and Lee (2021) organized the corpus, which presently stands as the most extensive dataset for sequential sentence classification. This dataset is sourced from the PubMed repository of biomedical literature, where each sentence within every abstract is assigned a label indicating its significance in the abstract. These labels fall within the categories of objective, background, conclusion, result, and method.

PubMed 200k RCT dataset distribution

We compile the dataset by extracting Randomized Controlled Trials from the PubMed database of biomedical literature. This dataset offers a standardized array of sentence labels, encompassing Background, Objectives, Methods, Results, and Conclusion.

The important distribution of the abstracts available from the PubMed 200k RCT is given in table 1. |L| denotes number of the labels, |V | represents vocabulary size. For the training, validation, and testing datasets, we indicate the number of the abstracts followed by the number of sentences in open parentheses.

Table 1: PubMed Dataset Statistics

Datasets	|L|	|V|	Training	Validation	Testing
PubMed 20k	5	68k	15k (180k)	2.5k (30k)	2.5k (30k)
PubMed 200k	5	331k	190k (2.2M)	2.5k (29k)	2.5k (29k)

III. Data Preprocessing

To handle the text data, we have to perform visualization so as to prepare the dataset for the deep learning models. We create a function designed to accept the file path of 21 a text file. This function then proceeds to open the file, read each line within it, and finally return the lines of text as a list.

Creating list of dictionaries containing Key/ Value pairs

We formulate an additional preprocessing function that accepts the filename as input. This function then reads the contents of the file and navigates through each line to extract pertinent details such as the target label, the sentence text, the count of sentences in the ongoing abstract, and the sentence number at which the target line is positioned. A collection of dictionaries is furnished, encompassing a line from the abstract, the corresponding line label, the line's position within the abstract, and the total count of lines within that specific

abstract. This function is harnessed to handle every text file within our dataset, including the train, validation, and test text files.

Numerical conversion of labels present in Abstract

For the conversion of labels in an abstract to numeric notation, we use the One Hot encoding and Label Encode Labels which convert the target columns of the label in an abstract into the form of [0,1] matrices for each label and array of integers respectively.

Data Preparation for deep multilevel embedding model

After the data preprocessing and numeric labels generation process, we need to prepare this data in order to feed into the deep models. For this, we perform the task of vectoring and developing of multiple embedding layers. The vectorization layer translates the textual content into numerical representations, while the embedding layer captures the interconnections among these numerical values. This process is facilitated by suitable libraries, specifically *TensorFlow* and *NumPy*.

IV. Models

Convolution 1D model with token embeddings

The convolution process is conducted to extract distinctive features from the input text. Convolutional layers are employed to derive fundamental semantic features from the initial textual data, simultaneously diminishing the dimensionality, in contrast to the sequential data processing by BiLSTM Jafari (2022). The use of 1D convolution kernels is implemented to perform convolutions across the input vectors.

In this context, the equation provided below formulates an array of sequential text by amalgamating the embedding vectors of individual words within it:

$$X_{1:T} = [x_1, x_2, x_3, x_4, \cdots x_T] \quad (1)$$

In this context, T denotes the count of tokens in the text. To encapsulate inherent attributes utilizing a 1D CNN, convolution kernels of diverse sizes are employed on Text Vectorization $X_{1:T}$. This aims to encapsulate the unigram, bigram, and trigram characteristics of the text. In the final convolution, when a d-word window spanning from $t{:}t{+}d$ is treated as input, the convolution procedure generates features for this window as depicted below:

$$h_{d,\,t} = \tanh\,(W_d x_{t:t+d-1} + b_d) \quad (2)$$

Here, $x_{t:t+d-1}$ denotes the embedding vectors of words within the window, Wd represents the adaptable and trainable weight matrix, and bd indicates the applied bias.

Model Architecture

We use layers.Conv1d from the *TensorFlow* library for the token embedding with activation function ReLU. 1D Global Average Pooling is used for condensing the output of the feature vector, with a dense layer used as output and activation function with *Softmax*.

Layer (type)	Output Shape	Param #
input_1 (InputLayer)	[(None, 1)]	0
text_vectorization (TextVectorization)	(None, 55)	0
token_embedding (Embedding)	(None, 55, 128)	8299648
conv1d (Conv1D)	(None, 55, 64)	41024
global_average_pooling1d (GlobalAveragePooling1D)	(None, 64)	0
dense (Dense)	(None, 5)	325

Figure 3: Architecture of the 1D CNN model for token level embeddings

Convolution 1D model with character level embeddings

This embedding design aims to address the limitations of both the bag of words and word-level embedding strategies. This method executes word embedding at the character level, disregarding the syntactical and semantic information within the input sentences. The character-level embeddings implemented with CNN provides the feasibility to work with several languages without the knowledge of the important meaning of the word from the dictionary. In the representation of character-based information, each input character is mapped to multidimensional spaces using 2-dimensional input tensors for text encoding. Consequently, the convolutional layer processes the input at the character level (Alharbi, and Lee, 2022).

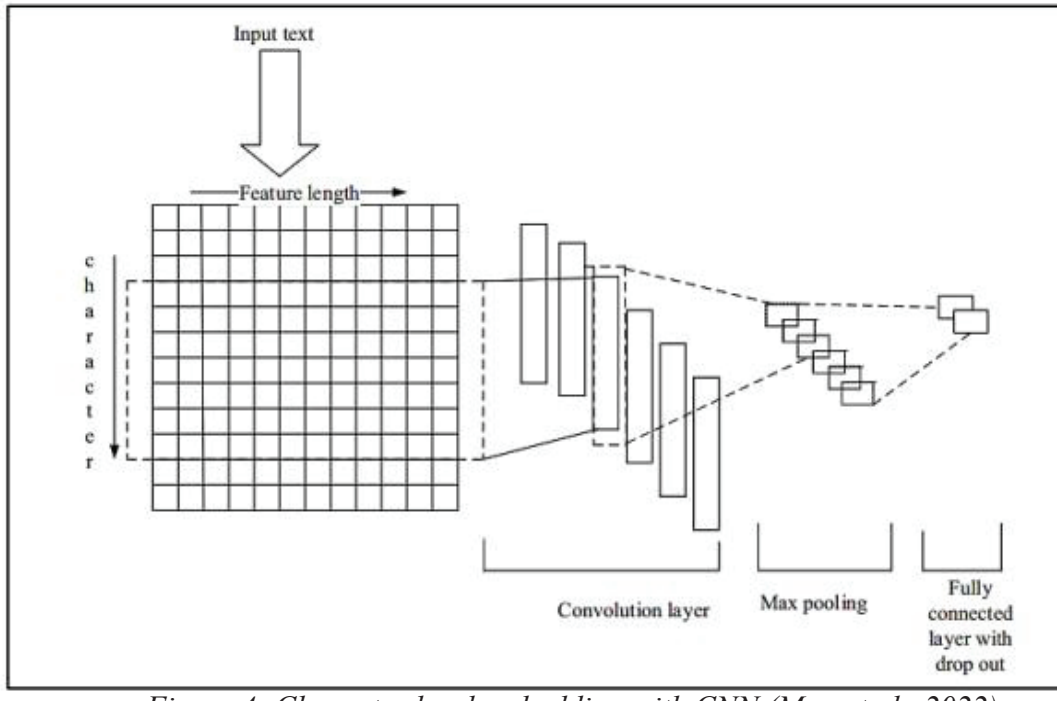


Figure 4: Character level embedding with CNN (Mao et al., 2022)

Input to the model is sequence of characters which is mapped to the character matrix $C \in R^{d \times |V|}$, where sentence matrix for the input words is defined as $S \in R^{d \times |S|}$. Forwarded in the layer of convolution which then performs the operation with the filter $F \in R^{d \times m}$, which is expressed by

$$b_j = (S * F)_j = \sum (S_{[:,j-m+1:j]} \otimes F)_{ki} \quad (3)$$

Here $S_{[:,j-m+1:j]}$ is The sentence matrix, along with its column size (m), is symbolized as a matrix, and the symbol $\otimes$ denotes element-wise multiplication within this matrix. The output b_j signifies the element-wise product of the filter and the column matrix in the sentence matrix. The above equation signifies the single filter convoluted with the sentence matrices in which using a set of filters the feature matrix is presented by $F \in R^{n \times (|S|-m+1)}$. Subsequently, a rectified activation function is applied, employing the ReLU operation to acquire feature maps featuring positive values. This methodology aids in training the network to produce precise and reliable outcomes. The activation process involves the utilization of weight matrices, and pooling operations are conducted to sum the values derived from the convolutional layer. Following this, the output from the pooling layer is directed to the fully connected soft-max layer. The probability distribution function for this layer is then articulated by:

$$P(y = j|x, v, b) = softmax_j(x^T w + b) = e^{xTW_J + bj} / \sum K\ k=1\ e^{XTw_k + bk} \quad (4)$$

Where the label w_k and b_k defines the weight vectors and the biasness value for the matrix.

Model Architecture

Token-level embeddings segment the dataset into tokens (words) and execute embeddings for each individual token. On the other hand, character embeddings disintegrate sequences into characters and generate a distinct feature vector for each character. The Embedding class is utilized, and for the character-level embedding layer, an input dimension and output dimension are specified. The input embedding dimension corresponds to the count of distinct characters within the character vocabulary.

```
Layer (type)                 Output Shape              Param #
=================================================================
input_3 (InputLayer)         [(None, 1)]               0

char_vectorizer (TextVector  (None, 290)               0
ization)

char_embed (Embedding)       (None, 290, 25)           1750

conv1d_1 (Conv1D)            (None, 290, 64)           8064

global_max_pooling1d (Globa  (None, 64)                0
lMaxPooling1D)

dense_3 (Dense)              (None, 5)                 325

=================================================================
Total params: 10,139
Trainable params: 10,139
Non-trainable params: 0
_________________________________________________________________
```

Figure 5: Architecture of the 1D CNN model for character level embeddings

Hybrid embedding Model

The hybrid token embedding layer involves both token embeddings and character embeddings. To put it differently, it constructs a layered embedding to depict sequences prior to transmitting them to the optimization layer for sequence labeling. We employ the Bi-LSTM to effectively capture extensive relationships among words that extend over significant distances. To comprehend a word within its context, we necessitate an embedding mechanism, and this is where the contextual embedding layer becomes pivotal. This layer includes Bidirectional Long-Short-Term-Memory (LSTM) sequences for sentence embedding and contextualization. A bidirectional-LSTM (bi-LSTM) is employed, encompassing both forward and backward LSTM sequences. The resultant output embeddings from these two directions concurrently encode information from preceding (backwards) and subsequent (forward) states.

Model Architecture

This phase takes the embedding vector of each token in a sentence from the word embedding layer as input and produces a vector s to encode the sentence. The sequence of embedding vectors undergoes initial processing through a bi-directional LSTM, resulting in a sequence of hidden states for a sentence containing N words, with each hidden state corresponding to a word. To formulate the final representation vector, a dense layer is utilized.

Layer (type)	Output Shape	Param #	Connected to
char_input (InputLayer)	[(None, 1)]	0	[]
token_input (InputLayer)	[(None,)]	0	[]
char_vectorizer (TextVectoriza tion)	(None, 290)	0	['char_input[0][0]']
universal_sentence_encoder (Ke rasLayer)	(None, 512)	256797824	['token_input[0][0]']
char_embed (Embedding)	(None, 290, 25)	1750	['char_vectorizer[1][0]']
dense_4 (Dense)	(None, 128)	65664	['universal_sentence_encoder[1][0]']
bidirectional (Bidirectional)	(None, 50)	10200	['char_embed[1][0]']
token_char_hybrid (Concatenate)	(None, 178)	0	['dense_4[0][0]', 'bidirectional[0][0]']
dropout (Dropout)	(None, 178)	0	['token_char_hybrid[0][0]']
dense_5 (Dense)	(None, 200)	35800	['dropout[0][0]']
dropout_1 (Dropout)	(None, 200)	0	['dense_5[0][0]']
dense_6 (Dense)	(None, 5)	1005	['dropout_1[0][0]']

Figure 6: Architecture for Hybrid embedding model

Tribrid Model (Character, Token and Positional Embedding Model)

The line number within an abstract and the overall count of lines within that abstract constitute the columns of preprocessed data that can serve as features. While these features are part of the training dataset, which can also be provided to the model as a positional embedding. In simpler terms, the positional embedding denotes the positional information of a sentence within an abstract. It can be incorporated as a feature engineering technique during the training of datasets.

Create Positional Embeddings

As the line number and total number of lines columns are already numerical, they are fed into the model without any modifications. To achieve this, we leverage the *tf.one_hot* utility. This function generates a tensor that has undergone one-hot encoding. It accepts an array (or tensor) as input, and the depth parameter determines the dimensionality of the resultant tensor.

Model Architecture

Layer (type)	Output Shape	Param #	Connected to
char_inputs (InputLayer)	[(None, 1)]	0	[]
token_inputs (InputLayer)	[(None,)]	0	[]
char_vectorizer (TextVectorization)	(None, 290)	0	['char_inputs[0][0]']
universal_sentence_encoder (KerasLayer)	(None, 512)	256797824	['token_inputs[0][0]']
char_embed (Embedding)	(None, 290, 25)	1750	['char_vectorizer[2][0]']
dense_7 (Dense)	(None, 128)	65664	['universal_sentence_encoder[2][0]']
bidirectional_1 (Bidirectional)	(None, 64)	14848	['char_embed[2][0]']
token_char_hybrid_embedding (Concatenate)	(None, 192)	0	['dense_7[0][0]', 'bidirectional_1[0][0]']
line_number_input (InputLayer)	[(None, 15)]	0	[]
total_lines_input (InputLayer)	[(None, 20)]	0	[]
dense_10 (Dense)	(None, 256)	49408	['token_char_hybrid_embedding[0][0]']
dense_8 (Dense)	(None, 32)	512	['line_number_input[0][0]']
dense_9 (Dense)	(None, 32)	672	['total_lines_input[0][0]']
dropout_2 (Dropout)	(None, 256)	0	['dense_10[0][0]']
token_char_positional_embedding (Concatenate)	(None, 320)	0	['dense_8[0][0]', 'dense_9[0][0]', 'dropout_2[0][0]']
output_layer (Dense)	(None, 5)	1605	['token_char_positional_embedding[0][0]']

Total params: 256,932,283
Trainable params: 134,459
Non-trainable params: 256,797,824

Figure 7: Architecture for Tribrid Embedding Model

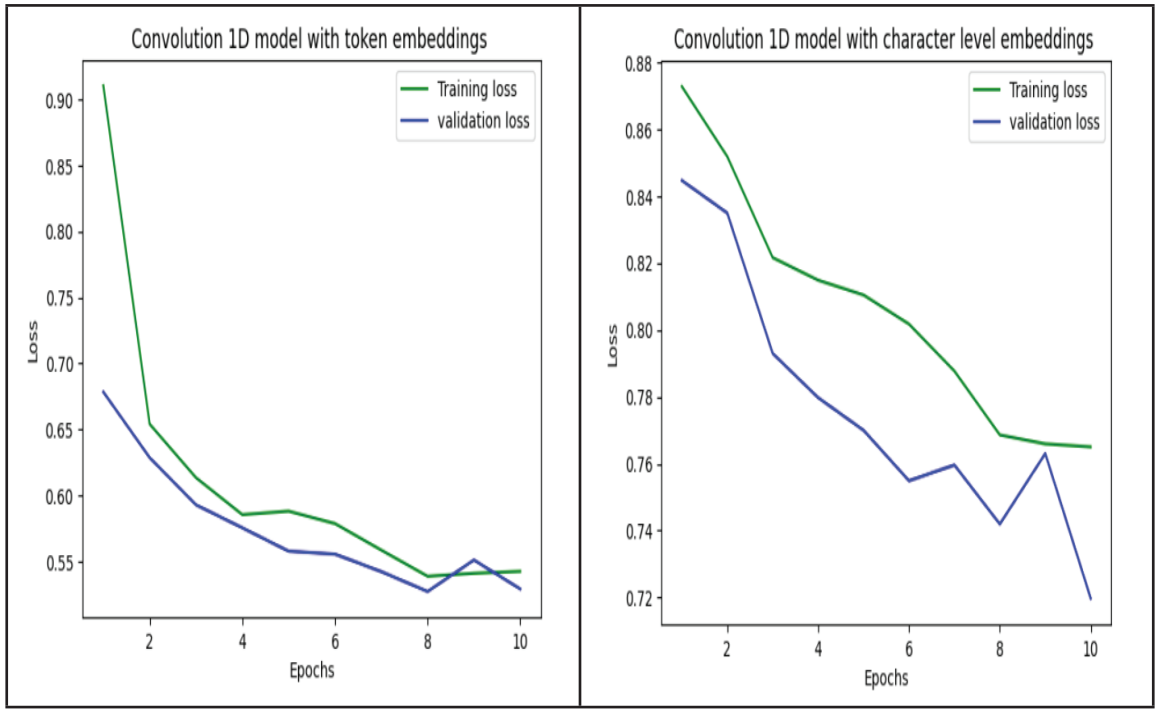
The Tribrid embedding model using the Bi-LSTM of character and token level embedding was further modified with the addition of a positional embedding feature in the hybrid

embedding model, which represents the exact position of a sentence for the particular abstract. Positional embeddings are generated to represent the position or order of the tokens within the sequence. This is done by assigning unique positional vectors to each position in the sentence. This type of embedding refers to the representation of tokens or words in a sentence with additional embeddings that encode their relative or absolute positions within the sequence. These embeddings help the model understand the sequential order and capture positional relationships among the tokens.

4. Results and Discussion

Fitting the Models

Since training contains nearly 200k lines of sentences, fitting the deep models took an ample amount of time. To make the experiments swift, we conducted training on a certain part of the training corpus, which is utilizing only 10 percent of the batches, amounting to approximately 18,000 samples, for training purposes. Additionally, the first 10 percent of the batches from the validation dataset were employed for validation across all the deep models.



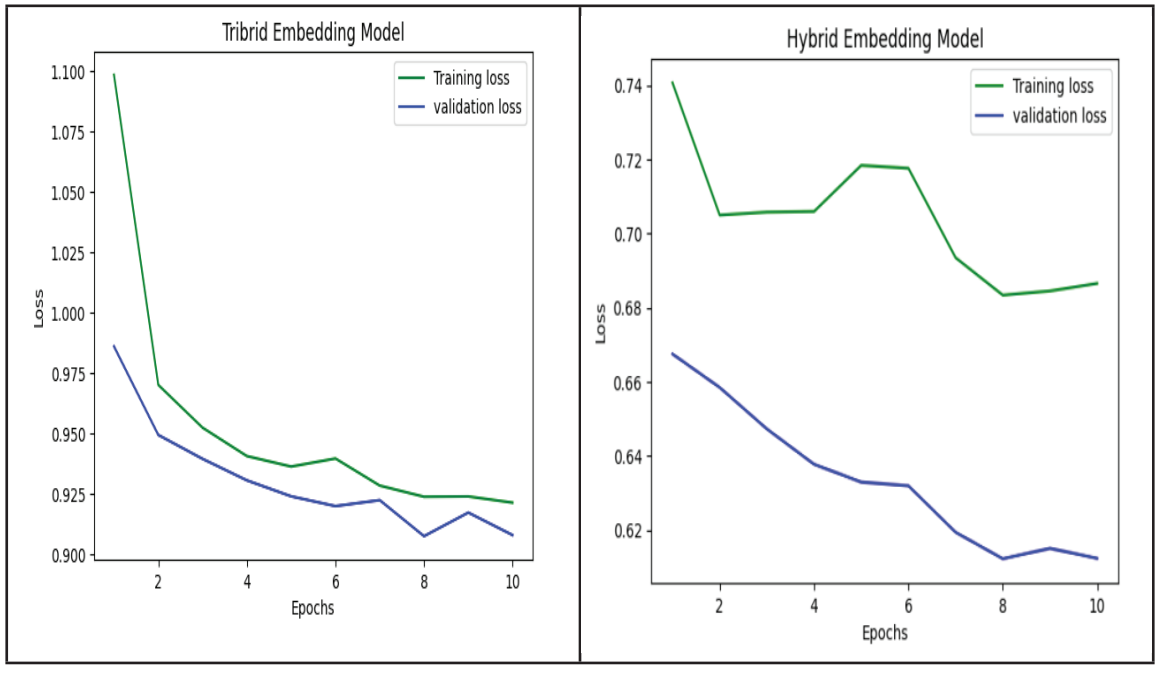


Figure 7: Loss for all models in PubMed 20k RCT dataset

The final result for all the models can be observed in the following table below. The Table 2 shows the second model using Convolutional 1D with character level embeddings achieved the lowest accuracy 71.686 and F1 score 0.710. Token level embedding model using also 1D CNN achieved better accuracy than model 2 with accuracy of 81.219. The third model employs BiLSTM with character level embedding and token embedding having the F1 score of 76.201 points which is slightly better than the previous model. Model 4 exceeded all the other models with the F1 score of 84.92.

Table 2: Performance Evaluation of all models

Metrics Model	Accuracy	Precision	Recall	F1 score
Model1:Conv1D with Token Embedding	81.219383	0.808039	0.812194	0.808891
Model2:Conv1D with Character Embedding	71.686747	0.709054	0.716867	0.710798
Model3:Hybrid Embedding	76.688071	0.764267	0.766881	0.762014
Model4:Tribrid Embedding	85.221104	0.853140	0.852211	0.849285

Model Comparison

The figure 9 shows the comparison of models based on the F1 score and all performance metrics respectively.

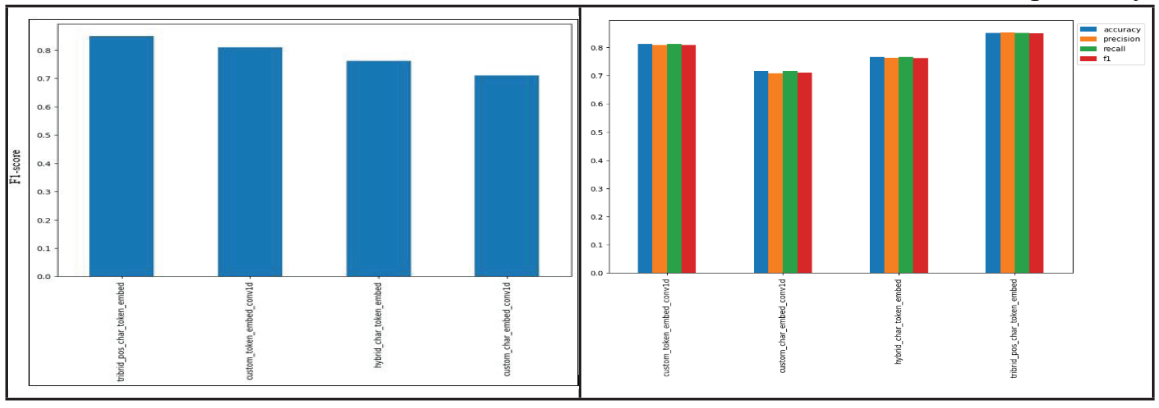


Figure 9: Model comparison based on all performance metrics based on F1 score and all performance metrics respectively.

Evaluation of test dataset on best performing Model (Tribrid Embedding Model)

To make the model's performance more comparable, we made predictions on the test dataset and evaluated them on our best performing model. Table below displays the outcomes of model evaluated on the available test dataset. The model provided the accuracy of 85.671 percent when tested with the test dataset of PubMed.

Table 3: Evaluation of Tribrid model on test dataset

Metrics	Result
Accuracy	85.671760
Precision	0.854261
Recall	0.855717
F1-score	0.854708

5. Conclusion and Future Enhancement

The work focuses on finding the enhanced method for sequentially classifying the sentences in a biomedical abstract. The method developed here implements the deep learning method with multilevel embedding approach where sentences of the biomedical abstract are sequentially classified under the semantic headings. We first examine each level in the reference model and mainly focus on Word embeddings method and the use of Attention mechanism and CRF layers. For each level of embeddings and multiple deep learning models, we propose the use of the Convolutional 1D and Bi-LSTM network. The

whole model is trained and evaluated using the PubMed RCT dataset. The model yields the best results for data validation in the Tribrid model with accuracy of 0.856 and least in the character level embedding model with accuracy of 0.716. This might be due to the limited number of training dataset used for character level embeddings. It is seen that the addition of positional information of the sentences for each abstract in the Tribrid model improves the sequential classification in a biomedical abstract. Further work can be done to improve the accuracy level of character level embedding process for sequential sentence classification by increasing the number of training and validating dataset available from PubMed RCT dataset. Also the performance of the system can be tested and analyzed using the other available dataset of Biomedical abstracts. Furthermore the architecture can be analyzed with varying parameters for the sequential sentence classification in the abstracts other than the biomedical background.

References

Arora, M., & Kansal, V. (2019). Character level embedding with deep convolutional neural network for text normalization of unstructured data for Twitter sentiment analysis. *Social Network Analysis and Mining*, *9*(1), 12.

Alharbi, A. I., & Lee, M. (2020, May). Combining character and word embeddings for the detection of offensive language in Arabic. In *Proceedings of the 4th Workshop on Open-Source Arabic Corpora and Processing Tools, with a Shared Task on Offensive Language Detection* (pp. 91-96).

Moudjari, L., Benamara, F., & Akli-Astouati, K. (2021). Multi-level embeddings for processing Arabic social media contents. *Computer Speech & Language*, *70*, 101240

Jin, D., & Szolovits, P. (2018). Hierarchical neural networks for sequential sentenceclassification in medical scientific abstracts. *arXiv preprint arXiv:1808.06161*

Firdaus, M., Golchha, H., Ekbal, A., & Bhattacharyya, P. (2021). A deep multi-task model for dialogue act classification, intent detection and slot filling. *Cognitive Computation*, *13*, 626-645.

Jang, B., Kim, M., Harerimana, G., Kang, S. U., & Kim, J. W. (2020). Bi-LSTM model to increase accuracy in text classification: Combining Word2vec CNN and attention mechanism. Applied Sciences, 10(17), 5841.

Dernoncourt, F., & Lee, J. Y. (2017). Pubmed 200k rct: a dataset for sequential sentence classification in medical abstracts. *arXiv preprint arXiv:1710.06071*.

Zhao, S., Su, C., Lu, Z., & Wang, F. (2021). Recent advances in biomedical literature mining. *Briefings in Bioinformatics*, *22*(3), bbaa057.

Tirotta, P., Yuasa, A., & Morita, M. (2023). Multilevel Sentence Embeddings for Personal-

ity Prediction. *arXiv preprint arXiv:2305.05748*.

Jafari, A. (2022). Comparison Study Between Token Classification and Sequence Classification In Text Classification. *arXiv preprint arXiv:2211.13899*.

Li, Q., Peng, H., Li, J., Xia, C., Yang, R., Sun, L., ... & He, L. (2022). A survey on text classification: From traditional to deep learning. *ACM Transactions on Intelligent Systems and Technology (TIST)*, *13*(2), 1-41.

Mao, K., Xu, J., Yao, X., Qiu, J., Chi, K., & Dai, G. (2022). A text classification model via multi-level semantic features. *Symmetry*, *14*(9), 1938.