Improved Sentence Classification For Medical Abstracts

Muhammad Hammad*
UET, Taxila
21SE30@uettaxila.edu.pk

Muhammad Yousuf Rana*
UET, Taxila
21SE58@uettaxila.edu.pk

May 14, 2025

Abstract

Current models for classification of medical abstracts lacks improved contextual understanding of the abstracts, leading to suboptimal outcomes. Our approach fuses the current architecture with the improved contextual understanding resulting in better classification of medical abstracts. This improved contextual understanding is the consequence of change in architecture, which now uses the advanced pre-trained model of Google Bert. By integrating the powerful pre-trained BERT model, our architecture gains a superior understanding of complex relationship and meaning within text. Thus more accurate and robust classification of medical abstracts is achieved, by this improved contextual understanding.

1 Introduction

An enormous amount of knowledge is contained within an ever-increasing corpus of scholarly articles. Bio-medical papers make up approximately half of the total, while other fields form the remainder of this vast knowledge reserve. [National Library of Medicine, 2025, 2023, Elsevier, 2025]. This increase in number has also propelled the increase in the heterogeneity of the bio medical literature, which is now not only limited to the scholarly article, but also includes clinical trial reports, electronic health records (EHRs), systematic reviews, and practice guidelines.

The norm for the researchers was to quickly skim through abstracts to find if paper is of interset to them, but this approach has lost its usefulness, due to main two reasons. One reason is the large amount of papers present now, which makes such skimming time-consuming and resource-intensive. Second reason is the unstructured nature inherent to many papers in which abstract is not classified by its author making information to be searched very difficult to find. If such large knowledge reserve is to be made useful, new methods of making use of abstract shall be employed.

This new method, making use of NLP, must classify the sentences in the abstract to appropriate headings like objectives, methods, results and conclusion, making finding useful information easier. This paper propose such method by making use of multi-input architecture that integrates information from three sources, dimensional contextual embeddings derived from a pre-trained Google BERT model, sequence representations learned from character embeddings via a Bidirectional LSTM, and explicit positional features indicating the line number and total number of lines within an abstract. These features streams are concatenated and fed into model for final classification. We evaluate our proposed model on PubMed RCT collection dataset.

*These authors contributed equally to this work.

2 Related Work

Automatic sentence classification is an important task in the field of biomedical natural language processing which is used in structured information extraction and other downstream tasks application such as evidence synthesis, clinical decision support, and biomedical information retrieval.

Early advancements in this field were driven by traditional machine learning approaches which included support vector machines [Kim et al., 2011, Liakata et al., 2012, Vapnik, 1998], Naive Bayes classifiers [Agarwal and Yu, 2009, Liakata et al., 2008, McCallum and Nigam, 1998], and random forest methods [Subramanian and Hval, 2016, Breiman, 2001]. These traditional methods relied on manually engineered text features such as term frequency–inverse document frequency (TF–IDF) vectors [Salton et al., 1975, Jones, 1972], n-grams [Brown et al., 1992, Shannon, 1948], part-of-speech tags [Church, 1988, Cutting et al., 1992], and shallow linguistic patterns [Hearst, 1992, Finn and Krause, 2006]—to capture discriminative cues for sentence role classification. While those approaches achieved respectable performance on benchmark datasets [Pang et al., 2002, Maas et al., 2011, Li and Roth, 2003], they still lacked in modeling longer-range semantic dependencies and subtle contextual nuances across sentences [Turney, 2001, Dasgupta and Ng, 2007, Řehůřek and Sojka, 2011]. This inherent limitation motivated the shift toward neural network models that can learn hierarchical and contextual features directly from the text [Collobert et al., 2011, Mikolov et al., 2013, Goldberg and Levy, 2014, LeCun et al., 2015, Bengio et al., 2003].

Recent progress in biomedical natural language processing has overwhelmingly favored deep neural network and transformer-based approaches. Franck Dernoncourt and Ji Young Lee [Dernoncourt and Lee, 2017] first employeed a hybrid CNN + LSTM architecture to the PubMed 200K RCT dataset, showing substantial increase in the accuracy. Shortly after Dernoncourt et al. [Dernoncourt et al., 2016] introduced a joint ANN + CRF model that optimized token and sequence level features simultaneously, resuting in improved consistency in section labeling. This deep learning based approaches were further accelerated by use of transformer based, domain specific pre-trained language models. SciBERT [Beltagy et al., 2019] makes use of unsupervised pretraining on specific texts to boost perfomance on biomedical NLP task by several F1 points. BioBERT furher refined this approach by pretraining on PubMed abstracts and full-text articles, acheiving state of art accuracy on sentence classification benchmarks. In more recent times, PubMedBERT was retrained from scratch , solely on PubMed data outperfoming other variants by 2-3

While these powerful architectures deliver near human perfection level, their need of large computational resources for training and inference, still remains a hindrance in low-compute resource environments. Our work presents a simple forward ANN architecture with an accuracy of 90% on PubMed dataset that is less resource expensive and can work in low-compute environments.

3 Model

3.1 Overview of the Architecture

We designed a multi-input neural network architecture to serve as computationally efficient model for sentence classification on the PubMed 200k RCT Dataset[Kim et al., 2011, Liakata et al., 2012].The model structure effectively integrates deiverese features including token-level embeddings from a pre-trained BERT model, character-level representations, and abstract-specific positional information(sentenece line number and total lines in the abstract). These inputs are processed through specialized branches and afterward combined and fed into subsequent dense layers to final classification.

3.2 Input representation

The model uses four distinct input representations for each sentence, processed in parallel branches.

3.2.1 Token Input

Raw sentence text is processed through BERT preprocessing layer and a pre-trained, frozen BERT model to generate a fixed-size pooled output vector representing the sentence semantic content.

3.2.2 Character Input

Sequence of characters is generated bt raw sentence text , which are then indiced by a TextVectorization layer. These indices are embedded using a learnable matrix $W_{\text{char\_emb}} \in \mathbb{R}^{(|\mathcal{V}_{\text{char}}|+1)\times d_c}$, where $|\mathcal{V}_{\text{char}}|$ is the size of the character vocabulary and d_c is the character embedding dimension.

For each position j, the character embedding vector is computed as

$$\mathbf{w}_j = W_{\text{char\_emb}}[\, x_j \,] \in \mathbb{R}^{d_c}.$$

This results in a sequence of character embedding vectors $\mathbf{W}_{\text{char}} = (\mathbf{w}_1, \ldots, \mathbf{w}_{n_c}) \in \mathbb{R}^{n_c \times d_c}$. This sequence can also be viewed as a matrix:

$$\mathbf{W}_{\text{char}} = \begin{bmatrix} \mathbf{w}_1 \\ \mathbf{w}_2 \\ \vdots \\ \mathbf{w}_{n_c} \end{bmatrix} \in \mathbb{R}^{n_c \times d_c}.$$

3.2.3 Total Line Number Input and Line Number Input

The total number of sentences in the abstract is represented as one hot encode vector and same is done for line number input to show sentence's relative positon.

3.3 Processing Branches

After the intial input representation , each representation is passed through a dedicated branch of the network to extract releveant features before they are evnetualy combined.

3.3.1 Token Processing Branch

The pooled output vector from the pre-trained BERT model, which represents the sentence high-level semantic features is processed through as sequence of two fully connected(dense) layers. RELU activation function is utilized in both dense layers. This branch synthesizes the representation suitable for combination with features of other branches.

The pooled output vector from the pre-trained BERT model is denoted as $\mathbf{v}_{\text{BERT}} \in \mathbb{R}^{d_{\text{BERT}}}$.

The first dense layer computes:

$$\mathbf{z}_1 = W_1 \mathbf{v}_{\text{BERT}} + \mathbf{b}_1$$

where $W_1 \in \mathbb{R}^{h_1 \times d_{\text{BERT}}}$ is the weight matrix, $\mathbf{b}_1 \in \mathbb{R}^{h_1}$ is the bias vector, and h_1 is the number of units in the first layer.

The activation is then applied:

$$\mathbf{a}_1 = \text{ReLU}(\mathbf{z}_1) = \max(0, \mathbf{z}_1)$$

The second dense layer computes:

$$\mathbf{z}_2 = W_2 \mathbf{a}_1 + \mathbf{b}_2$$

where $W_2 \in \mathbb{R}^{h_2 \times h_1}$ is the weight matrix, $\mathbf{b}_2 \in \mathbb{R}^{h_2}$ is the bias vector, and h_2 is the number of units in the second layer.

The final output of the token processing branch is:

$$\mathbf{a}_2 = \text{ReLU}(\mathbf{z}_2) = \max(0, \mathbf{z}_2)$$

3.3.2 Character Processing Branch

The sequence of character embeddings is moved through the Bidirectional Long Short-Term Memory(Bi-LSTM) network. The Bi-LSTM layers are made up of 256 units and are set to capture sequential and contextual information at the character level in both forward and backward direction along the sentence. This layer syntehizes a fixed-size vector representing aggregated character level features of the sentence.

For the forward pass, the LSTM computes hidden states $\vec{\mathbf{h}}_t$ for $t = 1, \ldots, n_c$:

$$\overrightarrow{\mathbf{h}}_t = \overrightarrow{\text{LSTM}}(\mathbf{w}_t, \overrightarrow{\mathbf{h}}_{t-1})$$

For the backward pass, the LSTM computes hidden states $\overleftarrow{\mathbf{h}}_t$ for $t = n_c, \ldots, 1$:

$$\overleftarrow{\mathbf{h}}_t = \overleftarrow{\text{LSTM}}(\mathbf{w}_t, \overleftarrow{\mathbf{h}}_{t+1})$$

At each time step t, the forward and backward hidden states are concatenated to form the combined hidden state $\mathbf{h}_t$:

$$\mathbf{h}_t = [\overrightarrow{\mathbf{h}}_t; \overleftarrow{\mathbf{h}}_t] \in \mathbb{R}^{2d_{\text{LSTM}}}$$

where d_{LSTM} is the dimensionality of each LSTM's hidden state (256 in this implementation).

3.3.3 Positional Feature Processing

The one-hot encoded positional features, namely the line number and the total line number in the abstract,are processed through the seperate, small feed-forward networks. Each of these input are connected to two dense layers with ReLU activation. This layer helps in learning the positional information.

3.3.4 Line Number Processing

The line number input is a one-hot encoded vector representing the position of the current sentence within the abstract.

Input: One-hot encoded vector $v_{\text{line}} \in \mathbb{R}^{D_{\text{line}}}$ (e.g., $D_{\text{line}} = 15$

The first dense layer computes:

$$z_{\text{line},1} = W_{\text{line},1} v_{\text{line}} + b_{\text{line},1}$$

where $W_{\text{line},1} \in \mathbb{R}^{128 \times D_{\text{line}}}$ and $b_{\text{line},1} \in \mathbb{R}^{128}$.

ReLU activation is applied:

$$a_{\text{line},1} = \text{ReLU}(z_{\text{line},1})$$

resulting in $a_{\text{line},1} \in \mathbb{R}^{128}$.

The second dense layer then computes:

$$z_{\text{line},2} = W_{\text{line},2} a_{\text{line},1} + b_{\text{line},2}$$

where $W_{\text{line},2} \in \mathbb{R}^{64 \times 128}$ and $b_{\text{line},2} \in \mathbb{R}^{64}$.

Applying ReLU activation again yields the output of the line number branch:

$$o_{\text{line}} = \text{ReLU}(z_{\text{line},2})$$

with $o_{\text{line}} \in \mathbb{R}^{64}$.

Total Line Number Processing

Similarly, the total line number input is a one-hot encoded vector representing the total number of sentences in the abstract. Input: One-hot encoded vector $v_{\text{total\_line}} \in \mathbb{R}^{D_{\text{total\_line}}}$ (e.g., $D_{\text{total\_line}} = 20$ as per the paper's example values from Source [48]).

The first dense layer computes:

$$z_{\text{total\_line},1} = W_{\text{total\_line},1} v_{\text{total\_line}} + b_{\text{total\_line},1}$$

where $W_{\text{total\_line},1} \in \mathbb{R}^{128 \times D_{\text{total\_line}}}$ and $b_{\text{total\_line},1} \in \mathbb{R}^{128}$.

ReLU activation is applied:

$$a_{\text{total\_line},1} = \text{ReLU}(z_{\text{total\_line},1})$$

resulting in $a_{\text{total\_line},1} \in \mathbb{R}^{128}$.

The second dense layer computes:

$$z_{\text{total\_line},2} = W_{\text{total\_line},2} a_{\text{total\_line},1} + b_{\text{total\_line},2}$$

where $W_{\text{total\_line},2} \in \mathbb{R}^{64 \times 128}$ and $b_{\text{total\_line},2} \in \mathbb{R}^{64}$.

Applying ReLU activation again yields the output of the total line number branch:

$$o_{\text{total\_line}} = \text{ReLU}(z_{\text{total\_line},2})$$

with $o_{\text{total\_line}} \in \mathbb{R}^{64}$.

3.4 Feature Combination and Classification

After processing through the respective branches, features extracted from the token, character and positional information(line number and total line numbers) are combined and passed through the subsequent layers for final classification.

The output vector from the token processing branch(derived from the BERT model) and the output vector from the character processing branch(the aggregated Bi-LSTM model output) are concatenated to form a combined feature vector, through use of the gating mechanism, Which is described below:

Following the independent processing in the token and character branches, which yield representations $v_{\text{BERT\_processed}}$ (from the token branch with $h_2 = 256$ units) and h_{char} (from the character BiLSTM, with

$2 \times$ lstm_units units), the model employs a mutual gating mechanism. This allows for a refined, context-aware fusion of semantic and character-level information.

1. **Character-to-Token Gating:** The character representation h_{char} is used to compute a gate $g_{c \to t}$ that modulates the token representation:

$$g_{c \to t} = \sigma(W_{ct} h_{\text{char}} + b_{ct})$$

where W_{ct} is a weight matrix for a dense layer (transforming h_{char} to dimension $h_2 = 256$), b_{ct} is its bias, and σ is the sigmoid activation function. The gated token representation is then:

$$v'_{\text{BERT\_processed}} = v_{\text{BERT\_processed}} \odot g_{c \to t}$$

where $\odot$ denotes element-wise multiplication.

2. **Token-to-Character Gating:** Similarly, the token representation $v_{\text{BERT\_processed}}$ computes a gate $g_{t \to c}$ for the character representation:

$$g_{t \to c} = \sigma(W_{tc} v_{\text{BERT\_processed}} + b_{tc})$$

where W_{tc} is a weight matrix for a dense layer (transforming $v_{\text{BERT\_processed}}$ to dimension $2 \times$ lstm_units), b_{tc} is its bias. The gated character representation is:

$$h'_{\text{char}} = h_{\text{char}} \odot g_{t \to c}$$

The two gated representations, $v'_{\text{BERT\_processed}}$ and h'_{char}, are concatenated:

$$f_{\text{tc\_gated}} = \text{concatenate}(v'_{\text{BERT\_processed}}, h'_{\text{char}})$$

The output of the dropout layer(representing the combined and processed token and character features), the output of the line number branch, the output of the total line number branch are concatenated to form the final feature vector.

To integrate the processed token-character representation d_{tc}, the line number representation o_{line}, and the total line number representation $o_{\text{total\_line}}$, the model utilizes an attention mechanism. This allows the model to dynamically assign importance to each feature stream when making a classification decision.

First, the three feature streams are concatenated:

$$f_{\text{pre\_attention}} = \text{concatenate}(o_{\text{line}}, o_{\text{total\_line}}, d_{\text{tc}})$$

Attention scores ($s_{\text{line}}, s_{\text{total\_line}}, s_{\text{tc}}$) are then calculated from $f_{\text{pre\_attention}}$, each having separate dense layer with a single output unit and linear activation:

$$s_{\text{line}} = W_{s,\text{line}} f_{\text{pre\_attention}} + b_{s,\text{line}}$$

$$s_{\text{total\_line}} = W_{s,\text{total\_line}} f_{\text{pre\_attention}} + b_{s,\text{total\_line}}$$

$$s_{\text{tc}} = W_{s,\text{tc}} f_{\text{pre\_attention}} + b_{s,\text{tc}}$$

These scores are normalized using a softmax function to yield attention weights $\alpha_{\text{line}}, \alpha_{\text{total\_line}}, \alpha_{\text{tc}}$:

$$[\alpha_{\text{line}}, \alpha_{\text{total\_line}}, \alpha_{\text{tc}}] = \text{softmax}([s_{\text{line}}, s_{\text{total\_line}}, s_{\text{tc}}])$$

Each of the original input feature streams to this stage ($o_{\text{line}}, o_{\text{total\_line}}, d_{\text{tc}}$) is then element-wise multiplied by its corresponding attention weight:

$$o'_{\text{line}} = \alpha_{\text{line}} \cdot o_{\text{line}}$$

$$o'_{\text{total\_line}} = \alpha_{\text{total\_line}} \cdot o_{\text{total\_line}}$$

$$d'_{\text{tc}} = \alpha_{\text{tc}} \cdot d_{\text{tc}}$$

Finally, these weighted feature vectors are concatenated together to form the comprehensive feature vector f_{final}:

$$f_{\text{final}} = \text{concatenate}(o'_{\text{line}}, o'_{\text{total\_line}}, d'_{\text{tc}})$$

For the combination of these features we use an attention mechanisim. This gives the model the ability to learn dynamically the contribution for each feature stream.

The final feature vector is passed into two more dense layers, both are using RELU function. The final output layers is a dense layer with five units corresponding to five target classes(BACKGROUND, CONCLUSIONS, METHODS, OBJECTIVE, RESULTS), each layer produces raw score(logit) corresponding to each class, which is used to compute loss during training.

Let f_{final} be the comprehensive feature vector from the attention mechanism. The first dense layer computes:

$$a_{\text{final},1} = \text{ReLU}(W_{\text{final},1} f_{\text{final}} + b_{\text{final},1})$$

where $W_{\text{final},1}$ and $b_{\text{final},1}$ are the weights and biases of this layer (e.g., transforming f_{final} to 256 units), and ReLU is the Rectified Linear Unit activation function.

The second dense layer takes $a_{\text{final},1}$ as input:

$$a_{\text{final},2} = \text{ReLU}(W_{\text{final},2} a_{\text{final},1} + b_{\text{final},2})$$

where $W_{\text{final},2}$ and $b_{\text{final},2}$ are the weights and biases of this layer (e.g., transforming $a_{\text{final},1}$ to 128 units). Finally, the output layer produces the logits for classification:

$$o_{\text{logits}} = W_{\text{output}} a_{\text{final},2} + b_{\text{output}}$$

where W_{output} and b_{output} are the weights and biases of the output layer, transforming $a_{\text{final},2}$ to C dimensions (the number of classes). These o_{logits} are then typically used with a softmax function during inference or a cross-entropy loss function (with logits) during training.

3.5 Model Compilation

The model is compiled using the categorical cross-entropy loss function. The categorical cross-entropy function is more stable than using a softmax activation on the output layer and then applying standard categorical crossentropy. Adam optimizer is used to optimize the training by using adaptive learning rate algorithim. During training and validation, the model perfomance is measured using the classification accuracy. The model was trained on two T4 GPUS using kaggle Notebook.

3.6 Model Summary

Figure 01 depicts the major processing branches of the model along with other parts like, dense layer and activations.

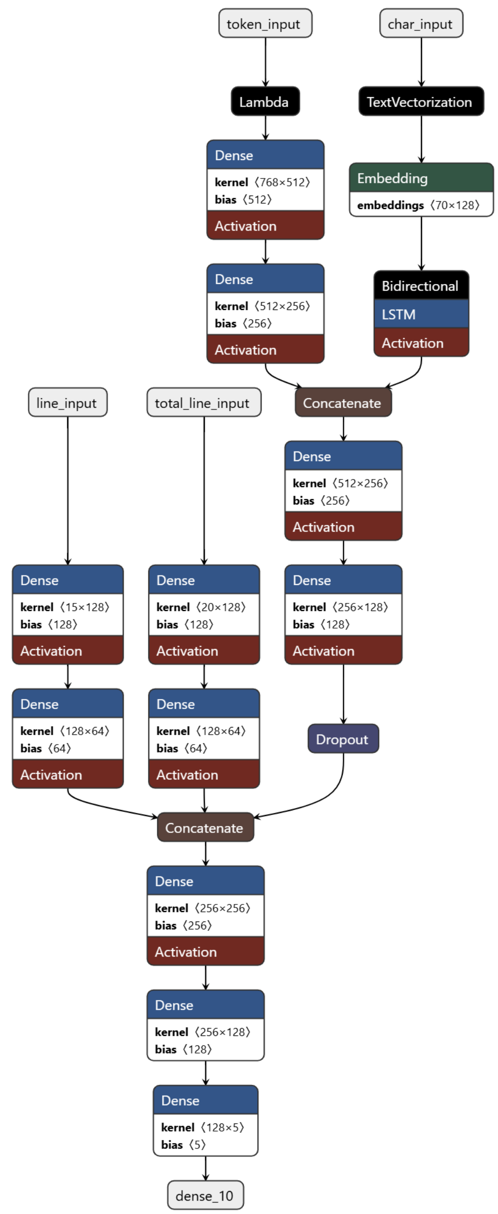

Figure 1: Model Summary.

4 Experiment

4.1 Dataset

We evaluate our model on the PubMed 20k RCT dataset, which is collection of randomized controlled trials (RCTs) from the PubMed database of biomedical literature, which provides a standard set of 5 sentence labels: objectives, background, methods, results and conclusions.

Table 1: Dataset Statistics

$\|C\|$	$\|V\|$	**Train**	**Validation**	**Test**
5	68k	15k (195k)	2.5k (33k)	2.5k (33k)

4.2 Training

The model is trained using Adam optimizer, which adapts learning rate method based on stochastic gradient descent. During training , the trainable parameters are updated for the goal of minimizing loss. The only paramters that are not updated are those assocaied with BERT because it is kept frozen.

For regularization to prevent overfitting, dropout is applied with a rate of 0.5. An additional dropout layer is added after the dense layer that processes concatenated output from the token and character branches, to capture more robust set of features.

Table 2: Performance on PubMed 20k

Model	**PubMed 20k**
LR	83.0
Forward ANN	86.1
CRF	89
Baseline model	89.3
Our model	**90.57**

Table 3: Classification Report

Label	**Precision**	**Recall**	**F1-Score**	**Support**
Background	0.75	0.85	0.80	3621
Conclusions	0.93	0.94	0.94	4571
Methods	0.94	0.95	0.94	9897
Objective	0.72	0.59	0.65	2333
Results	0.95	0.93	0.94	9713

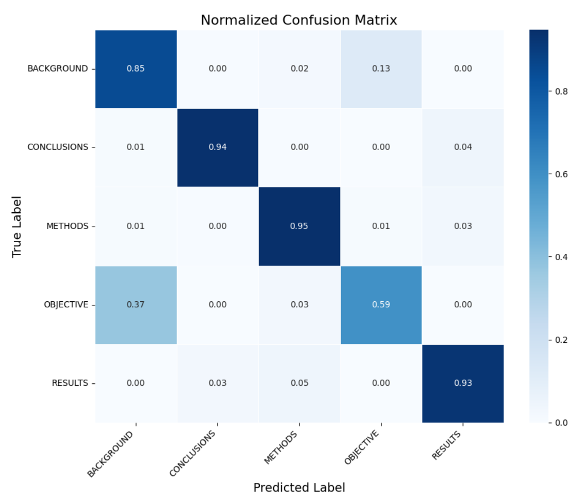

Figure 2: Confusion matrix.

5 Results and Discussion

Table 2 compares the following models, a simple logistic regression making use of n-gram features extracted from the current sentence and not using any surrounding sentences, the second model is the Forward-ANN model proposed by [Lee and Dernoncourt, 2016], a conditional random field employing n-grams feature making use of both preceding and current sentence when classifying the sentence, the third model presented by Lui [2012] that made use of a ANN with hybrid token embedding layer a sentence label prediction layer, and a label sequence optimization layer which acts as baseline for our paper, and finally our model proposed in the paper.

The reasons our model performs better than other models is because of

Leveraging Pre-trained Semantic Representations: The use of BERT model, fine tuned on PubMed Dataset, allows model to work with rich, context-aware token emdeddings. This method captures more nuance and detailed semantic relationship , that simple emdedding may miss.

Capturing Character-Level Nuances: Subword information is captured by model through use of character-level processing. This character level processing is useful in biomedical literature which often contains specialized terminologies, abbreviations and compound words.

Incorporating Positional Context: By making use of both line number and total line number in abstract, model learns to identify structural patterns in the abstract since most abstracts are structured in a specific way(for example Objective often comes first, Results in the middle, Conclusions at the end).

By combing high level semantic understanding , fine grained character details, and structural abstract positions, the model is better equipped to accurately classify sentences into their respective rhetorical roles within the PubMed RCT abstracts.

6 Conclusion

In this research paper, we propose a model making use of multiple features from token embeddings from a pre-trained BERT model, character-level embeddings generated by a Bidirectional LSTM, and positional features that capture sentence locations within an abstract. This multi-features learning of the model helps model in understanding nuance details and subtle patterns in the text, leading to better classification of sentences within the text. The effectiveness of this method is demonstrated by increase in accuracy as compared to the selected baseline.

References

Shashank Agarwal and Hong Yu. Automatically classifying sentences in full-text biomedical articles into introduction, methods, results and discussion. In *Bioinformatics*, volume 25, pages 3174–3180, 2009. doi: 10.1093/bioinformatics/btp548.

Iz Beltagy, Kyle Lo, and Arman Cohan. Scibert: A pretrained language model for scientific text. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, page 3613–3618. Association for Computational Linguistics, 2019. doi: 10.18653/v1/D19-1371. URL `https://aclanthology.org/D19-1371/`.

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. A neural probabilistic language model. *Journal of machine learning research*, 3(Feb):1137–1155, 2003.

Leo Breiman. Random forests. *Machine learning*, 45 (1):5–32, 2001.

Peter F Brown, Christophe Descloux, Robert L Mercer, Vincent J Della Pietra, Stephen A Della Pietra, Jen Chow Lai, and Joshua H Wright. Class-based n-gram models of natural language. *Computational linguistics*, 18(4):467–479, 1992.

Kenneth Ward Church. A stochastic parts program and noun phrase parser for unrestricted text. In *Proceedings of the second conference on Applied natural language processing*, pages 136–143, 1988.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, 12:2493–2537, 2011.

Doug Cutting, Julian Kupiec, Jan Pedersen, and Penelope Sibun. A hidden markov model part-of-speech tagger of english text. In *Proceedings of the 3rd conference on Applied natural language processing*, pages 112–119, 1992.

Sudeshna Dasgupta and Vincent Ng. A subsequence kernel for extracting sentence roles. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 671–679, 2007.

Franck Dernoncourt and Ji Young Lee. Pubmed 200k rct: a dataset for sequential sentence classification in medical abstracts. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, page 176–186. Association for Computational Linguistics, 2017.

Franck Dernoncourt, Ji Young Lee, Özlem Uzuner, and Peter Szolovits. Automatically identifying methodological sections of clinical research articles using deep learning. In *Journal of the American Medical Informatics Association*, volume 23, page 184–189. Oxford University Press, 2016.

Elsevier. Scopus: Abstract and Citation Database. `https://www.elsevier.com/products/scopus`, 2025. Accessed May 2025.

Adam Finn and Brigitte Krause. Recognizing question/answer pairs in online forums. In *Proceedings of the Human Language Technology Conference on Empirical Methods in Natural Language Processing*, pages 810–817, 2006.

Yoav Goldberg and Omer Levy. word2vec explained: deriving mikolov et al.'s skip-gram model from noise-contrastive estimation. *arXiv preprint arXiv:1402.3722*, 2014.

Marti A Hearst. Automatic acquisition of hyponyms from large text corpora. In *Proceedings of the 14th conference on Computational linguistics-Volume 1*, pages 539–545, 1992.

Karen Sparck Jones. A statistical interpretation of term specificity and its application in retrieval. *Journal of documentation*, 1972.

Su Nam Kim, David Martinez, Lawrence Cavedon, and Lars Yencken. Automatic classification of sentences to support evidence based medicine. *BMC Bioinformatics*, 12(Suppl 2):S5, 2011. doi: 10.1186/1471-2105-12-S2-S5.

Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436–444, 2015.

Ji Young Lee and Franck Dernoncourt. Sequential short-text classification with recurrent and convolutional neural networks. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, pages 515–520, San Diego, California, 2016. Association for Computational Linguistics.

Xin Li and Dan Roth. Learning question classifiers. In *Proceedings of the 17th International Conference on Computational Linguistics-Volume 1*, pages 362–368. Association for Computational Linguistics, 2003.

Maria Liakata, Simone Teufel, and Advaith Siddharthan. Identifying zones of conceptualisation in scientific texts. In *Proceedings of the 6th International Conference on Language Resources and Evaluation (LREC)*, pages 2053–2057, 2008.

Maria Liakata, Shyamasree Saha, Simon Dobnik, Colin Batchelor, and Dietrich Rebholz-Schuhmann. Automatic recognition of conceptualisation zones in scientific articles and two life science applications. *Bioinformatics*, 28(7):991–1000, 2012. doi: 10.1093/bioinformatics/bts071.

Marco Lui. Feature stacking for sentence classification in evidence-based medicine. In *Proceedings of the Australasian Language Technology Workshop 2012: ALTA Shared Task*, pages 134–138, 2012.

Andrew L Maas, Raymond E Daly, Peter T Pham, Dan Huang, Andrew Y Ng, and Christopher Potts. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 142–150, 2011.

Andrew McCallum and Kamal Nigam. A comparison of event models for naive bayes text classification. In *Proc. AAAI-98 workshop on learning for text categorization*, volume 752, pages 41–48. Citeseer, 1998.

Tomáš Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

National Library of Medicine. MEDLINE/PubMed Production Statistics. Technical report, U.S. National Library of Medicine, 2023.

National Library of Medicine. PubMed Overview. `https://pubmed.ncbi.nlm.nih.gov/about/`, 2025. Accessed May 2025.

Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. Thumbs up? sentiment classification using machine learning techniques. In *Proceedings of the ACL-02 conference on Empirical methods in natural language processing-Volume 10*, pages 79–86. Association for Computational Linguistics, 2002.

Radim Řehůřek and Petr Sojka. Gensim–statistical semantics in python. In *Proceedings of the Workshop on New Challenges for NLP Frameworks*, volume 37, pages 45–50, 2011.

Gerard Salton, A Wong, and CS Yang. A vector space model for automatic indexing. *Communications of the ACM*, 18(11):613–620, 1975.

Claude E Shannon. A mathematical theory of communication. *The Bell System Technical Journal*, 27(3):379–423, 1948.

Vaishnavi Subramanian and Michael Hval. Improving explanation generation for medical documents using random forest classifiers. *Journal of Biomedical Informatics*, 60:147–155, 2016. doi: 10.1016/j.jbi.2016.02.010.

Peter D Turney. Mining opinions from text using information extraction. pages 340–349, 2001.

Vladimir Vapnik. *Statistical learning theory*. Wiley New York, 1998.