SENTENCE CLASSIFICATION IN MEDICAL ABSTRACTS

MACHINE LEARNING FOR HEALTH CARE FS2022 APRIL 26, 2022

Frithiof Ekström **Kacper Kapuśniak** **Bartosz Rudnikowicz**

1 Introduction

An increasing number of Randomised Controlled Trials (RCT) are published every year [1]. Thus, it is more difficult for scientists and medical professionals to perform thorough systematic reviews - part of the job is to skim through the papers' abstracts to find their relevance. The structured abstracts, i.e. divided into sections, have been shown to help increase the efficiency of that process [2]. However, most of the published abstracts are still unstructured. The aim of this project was to automate the process of structuring abstracts by using the PubMed dataset [3] for the development of classification models assigning each sentence in the medical abstracts one of the classes: Background, Objective, Methods, Results, and Conclusions.

We first compared various classifiers and approaches to preprocessing (stemming, lemmatisation) using 'Term Frequency — Inverse Document Frequency' (TF-IDF). Next, we trained word embeddings - Word2vec [4], FastText [5], and Keras Embedding layer [6], and analysed their performance using 'advanced' models such as Bidirectional LSTMs and Residual Convolutional Networks. Further, we extended the classifiers architecture by extracting additional features about the position of the sentence in the abstract and Part-of-speech tagging. Finally, we tested the state of the art approach - pretrained Bidirectional Encoder Representations from Transformers (BERT).

2 Methods

2.1 Preprocessing and Baseline Models

2.1.1 Preprocessing

The PubMed dataset consists of abstracts, where every sentence is assigned one of five classes. So we started by preprocessing those sentences. Firstly, letters were lowercased, then stop words and punctuation were removed, and finally, all the numbers were replaced with the '@' sign.

Further, we explored normalisation methods - stemming and lemmatisation. Stemming replaces words with their stems - removes word endings by applying a certain set of rules. Lemmatisation maps the words into their lemma version from the dictionary and requires significantly more computing power, but typically yields better results than stemming. The goal of these methods is to reduce the size of the word space. The reduction might result in better embedding and classification performance, as we will be using the same vector for a set of similar words. Nonetheless, their main disadvantage is losing knowledge about the tenses used. For instance, backgrounds are likely to be written in past tenses, whereas objectives may also be written in present or future tense. [7]

More precisely, we used the Porter Stemmer algorithm, and WordNet Lemmatiser from the Natural Language Processing Toolkit [8, 9]. We decided to test three variants for baseline models - without any stemming or lemmatisation, only with stemming, and only with lemmatisation. We used the best performing option for the second task.

Finally, we retrieved the additional features for some of the 'advanced models' described in later sections. They include sentences' relative position in the abstract $\left(\frac{\text{sentence number}}{\text{abstract lenght}}\right)$ and the total number of sentences in the abstract. The main motivation here is a correlation between the class and its position in the abstract (e.g. background is at the beginning, and conclusions at the end). For the latter, there might be a correlation between the occurrence of the given class and the abstract length.

2.1.2 Baseline Models

For the baseline performance, we used TF-IDF as the sentence embedding. It measures the importance of the word to a document in a collection of documents by multiplying the word appearance number in a document with the inverse document frequency of a word across documents. [10] The model was configured to create a matrix corresponding to 1000 most frequent words to reduce the complexity. In our case, the sentences are treated as documents.

For the classification part, we compared Multinomial Naive Bayes [11], LightGBM [12] and Multilayer Perceptron (MLP). MLP consists of dense layers with sizes 64, 64, and 10 with a sigmoid activation function and 0.1 dropout. The detailed schematics can be found in 2a. To reduce the computational time for LightGBM (only), we selected the 150 best features from TF-IDF by computing the ANOVA F-value [13].

2.2 Advanced Embeddings and Models

2.2.1 Embeddings

Word2Vec and FastText are unsupervised methods to learn word embeddings. However, Word2Vec treats each word as an independent entity, while FastText sees each word as composed of character n-grams [4, 5]. Thus, FastText, in contrast to Word2Vec, can assign an informative vector to words unseen during training based on their n-grams. In the medical world, the new word formation is not uncommon; thus, there is an advantage here for the real-word application. However, the bias related to the imposed word similarity can also hinder the classification performance. Thus, we tested both embeddings for all our models to compare their performance. We set the size of the embeddings to 100.

Additionally, we tested the Embedding Layer in Keras [6] instead of using Word2Vec/FastText. It is trained at the same time as the classifier using backpropagation. The embedding size selected was again 100. The main advantage of that approach is that it is tuned

Table 1: Performance metrics for baseline models for various configuration of preprocessing

	No Stemming/Lemmatisation		Lemmatisation		Stemming	
	F1 (weighted)	**Accuracy**	**F1 (weighted)**	**Accuracy**	**F1 (weighted)**	**Accuracy**
Multinomial Naive Bayes	0.694	0.708	0.692	0.707	0.675	0.693
LightGBM	0.689	0.700	0.693	0.704	0.701	0.712
Multilayer Perceptron	0.769	0.775	**0.771**	**0.776**	0.769	0.774

for a particular task. However, since the learning is supervised, there is no way of creating embedding for words absent from the training dataset.

Finally, we extracted Part-of-speech (POS) tagging, which will be an additional input in some of our 'advanced' models. The tags assign a corresponding part of speech (such as a verb or noun) to each word in the text. The motivation for including these is the fact that some classes might contain more of the particular POS than others. For example, intuitively, conclusions might have more adjectives than other classes. POS tags were obtained using a tagger from the spaCy library [14] and encoded as TF-IDF features.

2.2.2 Models

First, we tested the above embedding variations (Word2Vec, FastText, and Keras layer) with the same MLP model as used for the baseline.

Next, we implemented an architecture consisting of two Bidirectional LSTM layers [15, 16]. The first LSTM layer takes as an input word embeddings from the text, and the second one takes the outputs of the first layer. The output of those layers is then passed to two dense layers with sizes 20 and 5.

Furthermore, we build upon the Bidirectional LSTM architecture to use the POS tagging features and abstract positional features. The former are passed into the separate, fully connected layer with 30 neurons. The output of this layer is concatenated with output from the last LSTM layer and positional information. The resulting tensor is then passed to the same two dense layers as in the unmodified architecture. The detailed schematics can be found in 2c.

Finally, inspired by the ResNet [17] architecture, we implemented a one-dimensional (1D) convolutional neural network (CNN) with residual connections. The architecture consists of a 1D convolutional layer, followed by two residual blocks and dense layers. In the first residual block, input and output have different numbers of channels; thus, the 'skip' connection consists of the additional convolutional layer. The second block does not change the number of channels, so no additional layer is needed. The output of the second block is flattened and concatenated with abstract positional information. Similar to previous models, the output of these layers is then passed to two dense layers with sizes 20 and 5. The detailed schematics can be found in 2b.

We fed the MLP model with sentence embeddings created by averaging word embeddings. For the Bidirectional LSTM and 1D ResNet, we used concatenated word embeddings and padding.

2.3 BERT

2.3.1 Base models

Two pretrained BERT [18] models were implemented for the final task. Due to the large domain-specific vocabulary of the dataset, we chose models that had been pretrained on speciality corpora of biomedical nature.

The Bio+Clinical BERT [19] model is an instance of BioBERT [20] trained on clinical text to perform named entity recognition and de-identification tasks. This model was chosen because, similarly to the article section prediction task, these tasks require both context-awareness and the capacity to handle domain-specific vocabulary.

The second model, SapBERT [21], uses PubMedBERT [22] as a base model and is additionally trained on a large compendium of biomedical vocabulary (UMLS 2020AA), to perform tasks involving the identification of medical concepts in articles. The rationale for including this model in our experiments was to explore the importance of vocabulary size in the pretraining, as the UMLS data is specifically designed to map a large vocabulary of medical terms to medical concepts.

2.3.2 Data preprocessing

The BERT models have their own preprocessing pipeline. Due to hardware constraints, only 100,000 samples, i.e. about 4.5% of the full training dataset, were used in training. The dataset was constructed by randomly drawing 20,000 samples from each of the five classes and without replacement. The data was then tokenized using Huggingface's fast tokenizer pretrained on the respective datasets of the models.

2.3.3 Training

A dense output layer of size 5 was added to each of the BERT base models, taking the pooled output of the base models as input. The models were first trained without tuning by freezing all layers except the output layer. Additionally, the models were tuned and trained with only the encoding layers frozen.

The models trained without tuning were trained for 10 epochs with a learning rate of 10^{-4} and a batch size of 8. However, due to the heavier nature of the task of training the models with unfrozen BERT layers, the tuned models were only trained for 1 epoch.

3 Results and Analysis

3.1 Preprocessing and Baseline Models

The results for the baseline models can be found in Table 1. It is evident that the Deep Learning (MLP) approach significantly outperformed the more conventional methods (Naive Bayes and Gradient Boosting). The large number of samples in the dataset (over 2 million sentences) can explain that - deep learning approaches are especially potent with larger datasets. However, the distinction is not as clear when analysing the differences between preprocessing methods. The Naive Bayes classifier performed the best without normalisation, while LightGBM and MLP variants correspondingly with Stemming and Lemmatisation. Further, the performance gaps between normalisations are not as significant as between the classification models. It might be justified by the fact that the reduction of words was marginal as medical datasets consist of many terminologies that cannot be grouped into a single stem or lemma.

Table 2: The most similar words according to doctor, pain, medicine according to Word2Vec and FastText (ordered by similarity)

Word	Word2Vec	FastText
doctor	physician, gps, nurse	physician, doctorheld, nonphysician
pain	discomfort, paindiscomfort, discomfortpain	pain@, painad, paint
medicine	herb, medica, recipe	medicina, medicinal, medici

Table 3: The closest word vectors to the results of 'Semantic equations' according to Word2Vec and FastText (ordered by similarity)

'Semantic Equation'	Expected	Word2Vec	FastText
cardiologist - heart + skin =	dermatologist	dermatologist, ophthalmologist	dermatopathologist, dermatologist
skin - dermatologist + ophthalmologist =	eye	eyelid, ocular	skinskin, foreskin

Thus, for Task 2, we decided to use the preprocessing pipeline with the best performance on the best baseline model - lemmatisation. However, it is essential to notice that TF-IDF is thoroughly different from word embeddings used in Task 2, so the best normalisation method with TF-IDF does not have to correspond to the best normalisation with Word2vec/FastText.

3.2 Advanced Embeddings and Models

3.2.1 Embeddings

We trained Word2Vec and FastText models and analysed the semantic relationship between the word embeddings created. Specifically, in Table 2 we demonstrate the vectors most similar to doctor, pain and medicine using cosine similarity. Then, in Table 3, we present the 'semantic equations', their expected result and the most (cosine) similar resulting vectors according to the embedding models.

The bias of FastText resulting from using words substrings is noticed. For instance, Word2Vec finds words similar to 'pain' correctly, while FastText incorrectly claims 'paint' is the 3rd most semantically similar. Furthermore, looking at vectors similar to 'doctor', the same pattern is spotted - FastText claims the 'nonphysician' to be the 3rd most similar, while it is a semantic antonym.

Both models handled the relationship 'a heart to the cardiologist is like a skin to the ?' well. For the 2nd query, we asked, 'a dermatologist to the skin is like an ophthalmologist to the ?'. Again, Word2Vec returned a satisfactory result, 'eyelid', while the 'skinskin' result as suggested by FastText was not as adequate ('skinskin' is likely an artefact created during preprocessing). Both observations reinforce the previous claim about the bias of FastText.

3.2.2 Models

We trained our models for each embedding with the 'Adam' optimiser [23] and early stopping. The results are presented in Table 4. It can be observed that the embedding types have little impact on models' performance and do not influence their ranking. The MLP achieves the best performance with the Keras embedding, the BiLSTM with the FastText embedding, and both models using positional features scored the best with Word2Vec. Further, all 'advanced' models beat the best baseline using TF-IDF, apart from MLP with FastText, which matched the performance.

However, the computational demand of the trainable Keras embedding layer is much higher than that of other models. Combined with the fact that it does not significantly improve performance compared to other embeddings, it is arguably the worst option. Surprisingly, the discrepancies noticed during the semantic analysis of Word2Vec and FastText, did not translate into the classification performance discrepancies.

Furthermore, the first and second-best models use positional features. These, along with POS tagging, boosts performance considerably - adding these features to the same BiLSTM architecture results in a performance increase of around 6-7% regardless of the embedding type, in terms of both accuracy and F1-score. It reconfirmed our correlation assumption between classes and features stated in the Methods section.

The confusion matrices are presented in Figure 1 for the best baseline model and all 'advanced' models with Word2Vec. Interestingly, Objectives and Results were rarely confused even for the baseline model. It might result from the fact that classes significantly differ in the occurrence of the numbers - in this case the '@' tag.

The most frequent misclassification for the baseline and models without additional features is between Background and Conclusions. However, it was almost eliminated by adding positional features, which justifies our decision to include positional features and part-of-speech tagging.

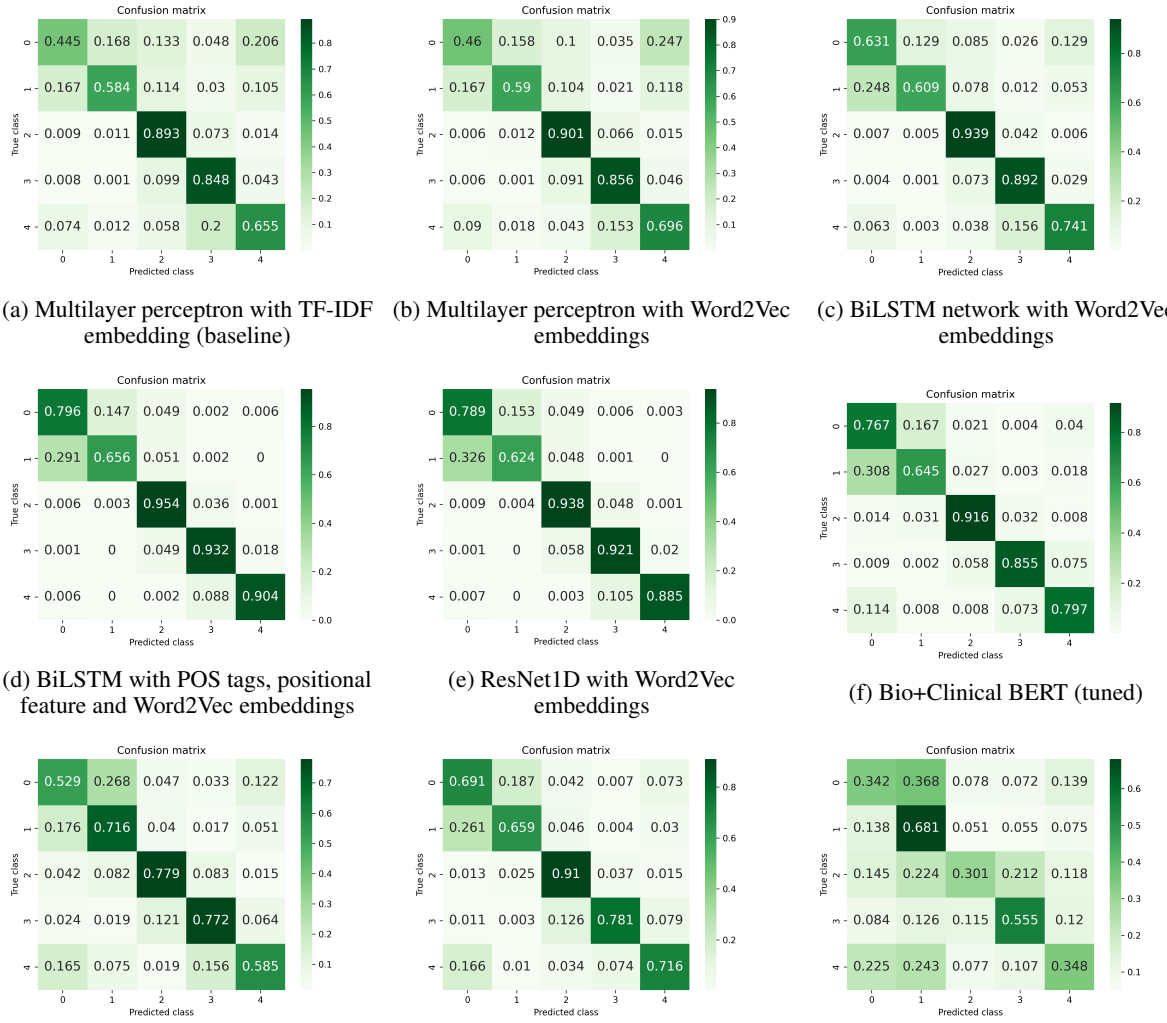
The subsequent common confusion for the baseline is Backgrounds/Objectives, and surprisingly, it is most prominent for the best performing models. The most probable explanation is that either class (and not both) is usually at the beginning of the abstract. In addition, positional features can help classify them as one of the two, but models have problems distinguishing between them. Thus, further research should focus on differentiation between Backgrounds and Objectives.

3.3 BERT

As can be observed in table 5, the BERT models reached similar levels of accuracy compared to the other models despite being trained on less than 5% of the data. It is illustrative of the power of transfer learning, enabling models to perform well on small datasets by utilizing information learned from a larger corpus used in pretraining.

Table 4: Performance metrics for models with various embeddings

	Word2Vec		FastText		Keras embedding	
	F1 (weighted)	Accuracy	F1 (weighted)	Accuracy	F1 (weighted)	Accuracy
Multilayer Perceptron	0.786	0.79	0.771	0.776	0.794	0.798
BiLSTM	0.836	0.839	0.839	0.841	0.825	0.826
BiLSTM with POS and position	**0.900**	**0.900**	0.899	**0.900**	0.899	**0.900**
ResNet1D	0.884	0.885	0.878	0.879	0.862	0.862



(a) Multilayer perceptron with TF-IDF embedding (baseline)

(b) Multilayer perceptron with Word2Vec embeddings

(c) BiLSTM network with Word2Vec embeddings

(d) BiLSTM with POS tags, positional feature and Word2Vec embeddings

(e) ResNet1D with Word2Vec embeddings

(f) Bio+Clinical BERT (tuned)

(g) Bio+Clinical BERT (untuned)

(h) SapBERT (tuned)

(i) SapBERT (untuned)

Figure 1: Normalised Confusion Matrices. Classes: Background (0), Objective (1), Methods (2), Results (3), Conclusions (4).

The Bio+Clinical BERT model outperformed the SapBERT model with a substantial margin. This likely follows from the kind of tasks that the models were pretrained on. SapBERT is pretrained on vocabulary data and was trained to detect medical concepts in texts. Such tasks primarily rely on the model's ability to connect specific words to concepts and may be less context-dependent than the named entity recognition task that the Bio+Clinical BERT model was trained on. Thus, the untuned model may lack the ability to pay attention to contextual words that are necessary for our task. It is also a potential explanation for the significant performance increase observed when tuning the model, as the unfreezing of attention layers alleviates such issues.

The BERT models also confused the background and objective samples the most. Since these models do not have access to the positional features of the 'advanced' models of the previous section, this suggests that these classes not only have a similar position in the abstract, but also have similar content. For instance, it could be the case that the background frequently mentions the objectives of previous studies.

4 Conclusion

In conclusion, our best model achieved accuracy and an F1 score of 0.900. We presented the dominance of the Bidirectional LSTM and the importance of additional feature extraction. We showed the superiority of deep learning models over some classical approaches. We also deduced that lemmatisation/stemming does not significantly impact the models' performance with TF-IDF embedding. Further, we demonstrated the bias in FastText introduced by n-grams and the fact that it has no impact on the classification performance. Furthermore, we revealed the capability of pre-trained BERT models - which could be unleashed with more computational resources. Finally, we stressed the importance of further research needed to classify Objectives and Backgrounds. Such models are pushing forward the area of sequential sentence classification. When applied in a real-world environment, it can significantly increase the work efficiency of researchers in the medical field.

Table 5: Performance metrics for BERT models

	Untuned		**Tuned**	
	F1 (weighted)	**Accuracy**	**F1 (weighted)**	**Accuracy**
Bio+Clinical BERT	0.727	0.720	**0.845**	**0.842**
SapBERT	0.447	0.431	0.798	0.796

References

[1] J. D. Niforatos, M. Weaver, and M. E. Johansen, "Assessment of publication trends of systematic reviews and randomized clinical trials, 1995 to 2017," *JAMA Internal Medicine*, vol. 179, no. 11, pp. 1593–1594, 2019.

[2] J. Hartley, M. Sydes, and A. Blurton, "Obtaining information accurately and quickly: Are structured abstracts more efficient?" *Journal of information science*, vol. 22, no. 5, pp. 349–356, 1996.

[3] F. Dernoncourt and J. Y. Lee, "Pubmed 200k rct: A dataset for sequential sentence classification in medical abstracts," *arXiv preprint arXiv:1710.06071*, 2017.

[4] T. Mikolov, K. Chen, *et al.*, "Efficient estimation of word representations in vector space," *arXiv preprint arXiv:1301.3781*, 2013.

[5] P. Bojanowski, E. Grave, *et al.*, "Enriching word vectors with subword information," *Transactions of the association for computational linguistics*, vol. 5, pp. 135–146, 2017.

[6] K. Ramasubramanian and A. Singh, "Deep learning using keras and tensorflow," in *Machine Learning Using R*, Springer, 2019, pp. 667–688.

[7] V. Balakrishnan and E. Lloyd-Yemoh, "Stemming and lemmatization: A comparison of retrieval performances," 2014.

[8] S. Bird and E. Loper, "Nltk: The natural language toolkit," Association for Computational Linguistics, 2004.

[9] C. J. Van Rijsbergen, S. E. Robertson, and M. F. Porter, *New models in probabilistic information retrieval*. British Library Research and Development Department London, 1980, vol. 5587.

[10] G. Salton and C. Buckley, "Term-weighting approaches in automatic text retrieval," *Information processing & management*, vol. 24, no. 5, pp. 513–523, 1988.

[11] C. Manning, P. Raghavan, and H. Schütze, "Introduction to information retrieval," *Natural Language Engineering*, vol. 16, no. 1, pp. 234–265, 2010.

[12] G. Ke, Q. Meng, *et al.*, "Lightgbm: A highly efficient gradient boosting decision tree," *Advances in neural information processing systems*, vol. 30, 2017.

[13] J. Tang, S. Alelyani, and H. Liu, "Feature selection for classification: A review," *Data classification: Algorithms and applications*, p. 37, 2014.

[14] M. Honnibal and I. Montani, "spaCy 2: Natural language understanding with Bloom embeddings, convolutional neural networks and incremental parsing," To appear, 2017.

[15] M. Schuster and K. K. Paliwal, "Bidirectional recurrent neural networks," *IEEE transactions on Signal Processing*, vol. 45, no. 11, pp. 2673–2681, 1997.

[16] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[17] K. He, X. Zhang, *et al.*, *Deep residual learning for image recognition*, 2015. DOI: 10.48550/ARXIV.1512.03385. [Online]. Available: https://arxiv.org/abs/1512.03385.

[18] J. Devlin, M.-W. Chang, *et al.*, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[19] E. Alsentzer, J. R. Murphy, *et al.*, "Publicly available clinical bert embeddings," *arXiv preprint arXiv:1904.03323*, 2019.

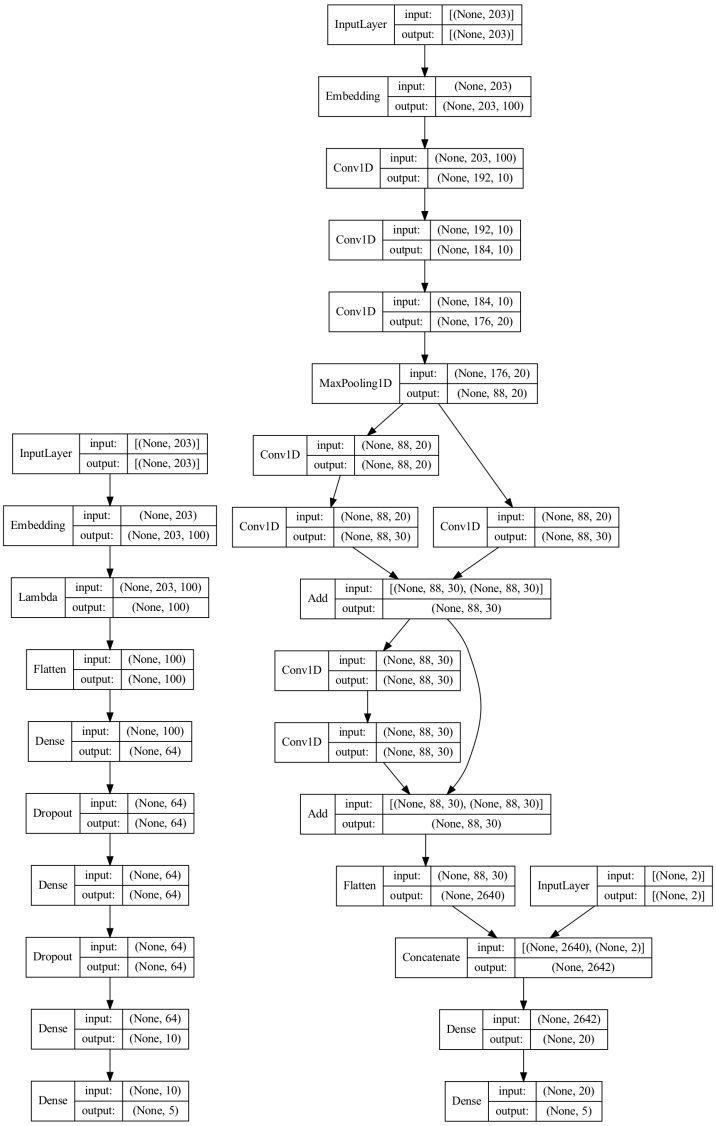
[20] J. Lee, W. Yoon, *et al.*, "Biobert: A pre-trained biomedical language representation model for biomedical text mining," *Bioinformatics*, vol. 36, no. 4, pp. 1234–1240, 2020.

[21] F. Liu, E. Shareghi, *et al.*, "Self-alignment pretraining for biomedical entity representations," in *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, Online: Association for Computational Linguistics, Jun. 2021, pp. 4228–4238. [Online]. Available: https://www.aclweb.org/anthology/2021.naacl-main.334.

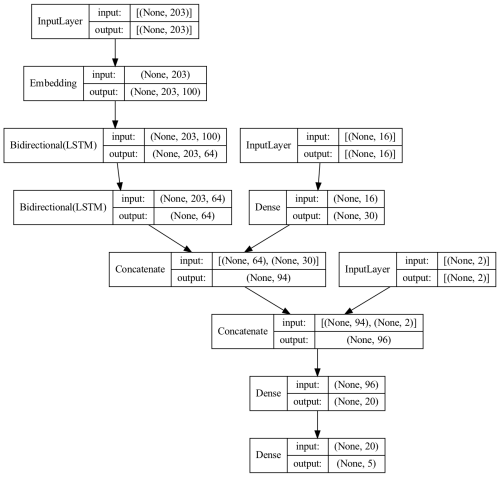
[22] Y. Gu, R. Tinn, *et al.*, *Domain-specific language model pretraining for biomedical natural language processing*, 2020. eprint: arXiv:2007.15779.

[23] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

Appendix A Models' Schematics



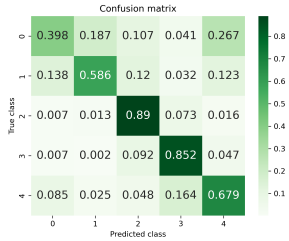
(a) MLP (b) ResNet1D



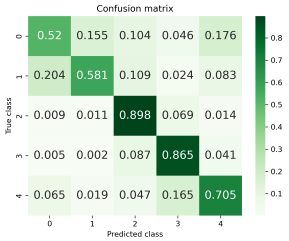
(c) Bidirectional LSTM

Figure 2: Models' Schematics

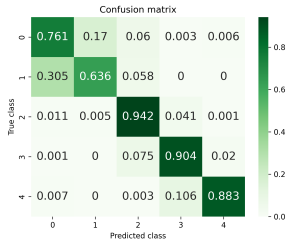
Appendix B Supplementary Confusion Matrices



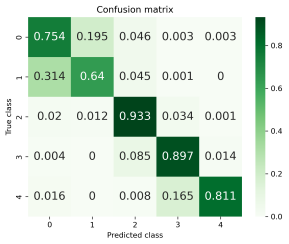
(a) MLP with FastText embedding



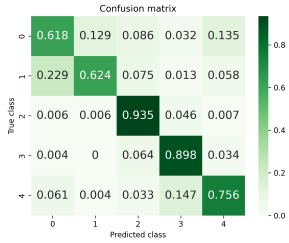
(b) MLP with Keras Embedding Layer



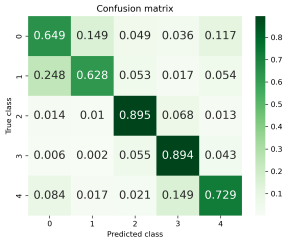
(c) ResNet1D with FastText embedding



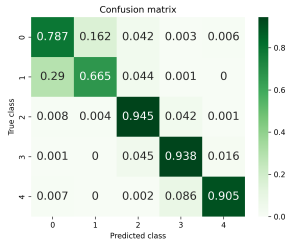
(d) ResNet1D with Keras Embedding Layer



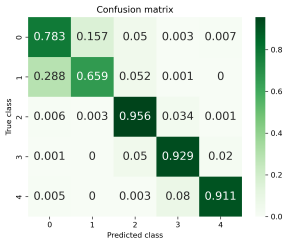
(e) BiLSTM with FastText embedding



(f) BiLSTM with Keras Embedding Layer



(g) BiLSTM with POS and positional features - FastText embedding



(h) BiLSTM with POS and positional features - Keras Embedding Layer

Figure 3: Supplementary normalised Confusion Matrices (FastText and Keras Layer embedding). Classes: Background (0), Objective (1), Methods (2), Results (3), Conclusions (4).