CS4248 Final Report

A0259561W, A0275725W, A0238146A, A0275766L, A0253860A
Team 04
Mentored by Xian He
{e0970533,e1127379,e0772744,e1127420,e0952398}@u.nus.edu

Abstract

Citations are an important part of academic documents as they help to establish the relevance of a new work relative to existing publications. Each citation has an intent, whether cited as prior literature, a methodology, or a result that grounds one's claims. Automated classification of citation intent helps in creating automated systems that streamline literature review and generate recommendations for further research. In our study, we examined the SciCite dataset and its features, utilized traditional feature engineering and machine learning techniques, and incorporated advanced methods such as Long Short-Term Memory (LSTM) and Bidirectional Encoder Representations from Transformers (BERT). Our research delved into feature processing and word embedding optimizations, with the LSTM combined with Self-Attention mechanism achieving the best performance, evidenced by a macro F1 score of 0.84. The primary goal of this research is to improve the prediction of citation intent and, more importantly, to discover methods for extracting more valuable information from similar datasets. This information can then be strategically applied to improve models for analogous machine learning tasks.[1]

1 Introduction

Recently, the exponential growth of academic paper text data demands enhanced information classification and processing. While traditional text classification methods offer reliable support in basic tasks, they often fall short when dealing with complex and high-dimensional datasets. Hence, this research strives to enhance the accuracy of academic paper text classification. Additionally, it seeks to uncover more insights from similar datasets and efficiently integrate this information into suitable models to bolster machine learning tasks.

The object of our study is the SciCite dataset(Cohan et al., 2019), an exhaustively annotated dataset containing citations of academic papers. This dataset contains not only traditional textual information, but also key metadata such as "sectionName", "isKeyCitation", etc., which may influence the classification results. Our objective is to predict the citation intent of a citation instance, into the classes of "background", "method" and "result".

This study first used traditional machine learning methods (such as BoW,TF-IDF vectorization plus LR, random forest and XGBoost) to process text data and obtained baseline performance (77%).To further improve performance, we introduce deep learning models based on BERT and LSTM, which are better able to handle long sequences of text and complex feature combinations (best macro F1 score is 84%).

We propose a universal research process aimed at continuously optimizing models through systematic analysis and experimental verification of data sets. This process includes steps such as data preprocessing, feature extraction, model training, and result evaluation, with special emphasis on the strategy of continuously improving the model based on feedback from experimental results. Additionally, we look into varying the word embeddings and modelling a more traditional LSTM approach to match the performance of newer transformer architectures.

The main contributions of this study include:

1. The influence of key features in academic document classification is revealed, providing a methodological basis for understanding and extracting useful metadata.

2. Adopting the different deep learning techniques, develop iterative improvements on the model, focusing on the different aspects that are important in considerations.

[1]Github link to our repository: https://github.com/CS4248-Team4/SciCite-Team4.

3. Investigation on the influence of word embedding model, directional dependencies and self-attention mechanism on the model's performance.

2 Related Work

Automatic classification of academic literature is a long-standing and active research topic in the field of natural language processing. Past research has mainly focused on how to use traditional machine learning techniques, such as Naive Bayes(Gan et al., 2021), support vector machines, and decision trees(Shah et al., 2020), to deal with text data classification problems. These methods have been widely used in the literature with some success. For example, Joachims (Joachims, 1998) discussed in detail the effectiveness of SVM in text classification tasks in his study. However, these traditional methods usually only consider the surface features of the text and ignore the deep semantic structure of the text.

With the rise of deep learning technology, researchers have begun to explore the use of complex models to understand and classify the deep features of text. In this regard, pre-trained language models such as BERT (Devlin et al., 2019) and LSTM (Hochreiter and Schmidhuber, 1997) have demonstrated their superior performance in various NLP tasks, including document classification. By learning linguistic regularities in large amounts of text data, these models are able to capture deeper semantic relationships than traditional methods.

In addition, some studies also focus on the impact of specific features on classification tasks. For example, Lai et al. (2015) explored how to improve text classification through ensemble learning methods in their paper, and their research showed that combining multiple features can significantly improve classification accuracy. Such research provides us with insights into how to integrate metadata such as "sectionName" and "isKeyCitation" in our models.

Although existing research has made some progress, there are still challenges in how to apply these advanced techniques to broader academic document classification tasks, especially when considering specific application scenarios such as classifying academic citations. Therefore, our research not only focuses on the application of technology, but is also committed to exploring a universal research process to continuously optimize the model through experimental feedback to adapt to changing data characteristics and needs.

3 Corpus Analysis & Method

The first step to building the classification models is to understand the dataset given. The SciCite dataset contains a wide range of labeled data of scholarly citations extracted from scientific articles, in fields such as computer science, bio-medicine. The contents of the data columns are as described in the appendix (Table 11).

In the dataset, the most important columns are the string and label columns, which provide the fundamental data that is needed, that is x, the contents to be trained over or predicted for, and y, the label associated with the content.

The string content in the training file is used for a self-trained word2vec model. This model is then used for vectorization, transforming the sentences into a sequence of 100-dimensional word vectors, which will then be fed into the classification model for processing.

The labels associated with the string are mapped to indexes respectively as shown in Table 1. Label2 is not considered in this case as it is a subset of the "result" label, having values of "supportive" and "not_supportive" which is not needed for the task at hand.

Index	Label
0	"background"
1	"method"
2	"result"

Table 1: Label mapping

In addition to these two primary features extracted from the dataset, exploring supplementary features can enhance the training and testing processes. These additional features contribute valuable insights to the learning or prediction. This feature engineering is mainly done by first identifying the possible features in the dataset, then analyzing these features to see whether there is any relationship present between them and the label.

Following the steps, we have identified several features which could aid in the classification task.

3.1 Additional Features

The 'sectionName' column serves as a vital indicator, as certain sections exhibit a pronounced tendency for their contents to align with specific categories. These section names are standardized by extracting keywords and categorized into one of the 14 categories detailed in Table 2. The analysis of

its relationship with the training dataset is depicted in Figure 1.

Index	Section
0	"discussion"
1	"introduction"
2	"unspecified"
3	"method"
4	"results"
5	"experiment"
6	"background"
7	"implementation"
8	"related work"
9	"analysis"
10	"conclusion"
11	"evaluation"
12	"appendix"
13	"limitation"

Table 2: Section name mapping

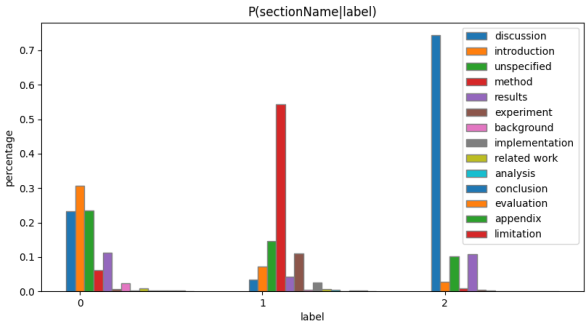

Figure 1: P(sectionName|label)

The distribution depicted in Figure 1 reveals that the probabilities of section names for each label are concentrated around a few specific options. For example, the section name 'discussion' prominently represents label 2.

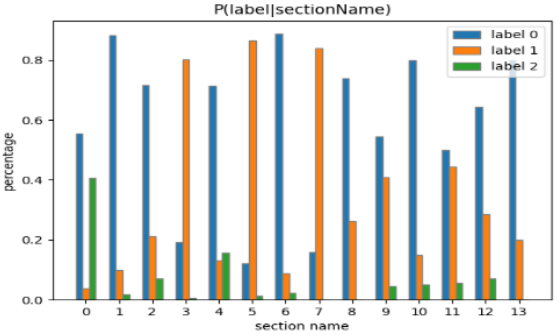

Figure 2: P(label|sectionName)

At the same time, when looking at the label distribution for each section name in Figure 2, it can be observed that there is a higher tendency for a certain section name to fall under a certain label.

Another feature identified to be useful is the 'isKeyCitation' column. The true value is mapped to 1 and false mapped to 0. Similarly to the section names analysis, the distribution for key citation and label is shown below in Tables 3 and 4.

There are also many other features that are identified from the dataset, while only the section name

Label	0	1	2
True	0.7	0.2	0.09
False	0.49	0.34	0.17

Table 3: P(label|isKeyCitation)

Label	True	False
0	0.53	0.47
1	0.32	0.68
2	0.3	0.7

Table 4: P(isKeyCitation|label)

and key citation feature are deemed to be useful as it presents a clear relationship with the labels, providing additional useful context to the string in the classification task process. While for the other features identified such as the number of citations in the string, the citation length, the string length, the position of citation in string and the label confidence, there is no clear distinction of relationship between the feature and the different labels observed (Figure 3) and is therefore not chosen to be used in the subsequent stages.

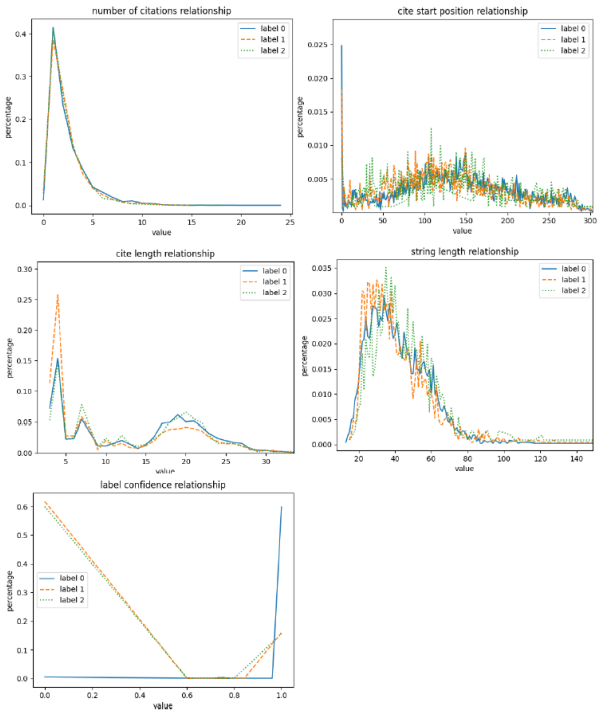

Figure 3: Percentage Distribution for the Other Features

4 Experiments

4.1 Traditional Experiments

In the early stages of this study, we explored the classification performance of a variety of traditional machine learning models on a dataset of academic literature. Table 5 shows the F1 scores of different model and feature engineering combinations. Specifically, we considered logistic regression (LR), support vector machine (SVM), naive Bayes (NB), random forest (RF), XGBOOST, and

CATBOOST models, combined with the bag-of-words model (BOW), TF -IDF and Word2vec as feature extraction methods.

Although these traditional methods show good performance under certain combinations, limitations in their structure and input characteristics lead to a performance ceiling. These models cannot effectively exploit structured features in the dataset, such as text sequence order and features such as "sectionName" and "isKeyCitation", which are critical for understanding text and classification. This finding suggests that while these traditional methods are capable of handling the surface level of text data, they are limited in their ability to capture deep semantic and structured information. Therefore, we conclude that more advanced models need to be explored to improve classification performance. Specifically, deep learning models, such as BERT and LSTM, are not only able to understand long sequences of text, but also integrate these complex features for classification. This drives our further research direction, which is to apply these advanced models to achieve in-depth analysis and more accurate classification of academic paper datasets.

Table 5: Traditional model & F1

MODEL	Feature engineering	F1 SCORE
LR	BOW	0.6925
	TF-IDF	0.6756
	Word2vec	0.6676
SVM	BOW	0.7211
	TF-IDF	0.6977
	Word2vec	0.5054
NB	BOW	0.7556
	TF-IDF	0.6943
RF	BOW	0.76419
	TF-IDF	0.7665
	Word2vec	0.6058
XGBOOST	BOW	0.77
	TF-IDF	0.7640
CATBOOST	BOW	0.7561
	TF-IDF	0.74812
	Word2vec	0.6358

4.2 LSTM Experiments with Features

We move on to investigate the ability of the LSTM architecture in comprehending both semantic and structured features in the dataset.

From our findings in the corpus analysis, we chose to experiment the effect of including each of these features into a uniLSTM model.

- String: Utilized a pre-trained word2vec to transform sentences into sequences of 100-dimensional vectors. Sequences were truncated to the average sentence length of 33, resulting in each sample being represented by a two-dimensional matrix of size [33,100].

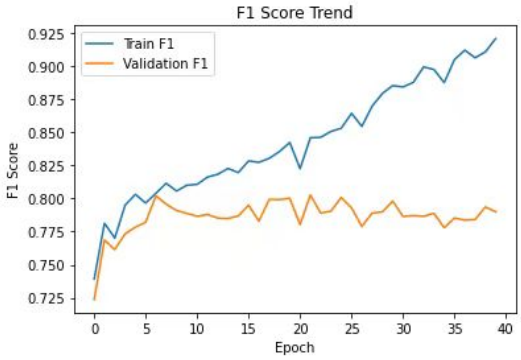

Figure 4: F1 Score Trend

- SectionName, label_confidence and isKeyCitation: After binning the section names into 14 classes as in Table 2, they were one-hot encoded into a 14-dimensional vector for each sample. We concatenate this with label_confidence and one-hot encoded isKeyCitation to form a single one-dimensional vector.

4.2.1 Model Structure

The processed string (vector sequence format) was first input into an LSTM. Following the LSTM output layer, the one-dimensional structured features vector was concatenated to this output, forming a single vector that was then fed into a fully connected layer and subsequently to a softmax layer for classification. If the last three features were not used, the concatenation process was omitted. We implemented a dropout and recurrent dropout rate of 0.2 in the LSTM layer to reduce overfitting. Training epochs was set at 20, at which we observe plateauing of the F1 score on the validation set.

4.2.2 Result of LSTM

The baseline unidirectional LSTM model, utilizing solely the String feature, laid a foundational benchmark with a Macro F1-score of 0.75. This outcome is a testament to the strength of the LSTM model in processing pre-trained Word2Vec word embeddings, which effectively capture semantic information from text even in the absence of additional features.

The introduction of the SectionName feature brought about a significant uptick in the model's performance. An increase in the Macro F1-score to 0.82 suggests that incorporating contextual information pertaining to the categorization of sections provides the model with a more nuanced understanding of the text.

Model	Feature	Macro Precision	Macro Recall	Macro F1	Micro F1
LSTM	String	0.77	0.74	0.75	0.78
LSTM	**String, sectionName**	**0.83**	**0.81**	**0.82**	**0.84**
LSTM	String, sectionName, isKeyCitation	0.82	0.81	0.81	0.83
LSTM	String, sectionName, isKeyCitation, label_confidence	0.50	0.35	0.26	0.55
BiLSTM	**String, sectionName**	**0.83**	**0.82**	**0.82**	**0.84**

Table 6: Feature Selection on Model Performance (averaged across 5 runs)

Model	Feature	Macro Precision	Macro Recall	Macro F1	Micro F1
BERT	String	0.74	0.78	0.75	0.78
BERT	String, sectionName	0.73	0.79	0.73	0.75
SciBERT	**String, sectionName**	**0.77**	**0.85**	**0.79**	**0.82**
SciBERT (SMOTE)	**String, sectionName**	**0.81**	**0.84**	**0.82**	**0.84**

Table 7: Feature Selection on Model Performance (averaged across 5 runs)

Incorporating label_confidence and isKeyCitation turned out to be counterproductive. The decline in the Macro F1-score to a mere 0.26 raises questions about the quality and representation of these features. Moreover, the drop in sample size from over 8000 to 6000 due to missing label_confidence data likely contributed to the model's under-performance. A reduced dataset may not capture the full variability and complexity of the problem space, leading to poor generalization to unseen data. This effect is particularly pronounced in smaller categories such as 'result', which the model failed to classify correctly at all.

Lastly, we compare the effect of swapping out the UniLSTM layer with an equivalent BiLSTM layer. The performance of both models are very similar, suggesting that capturing bidirectional dependencies does not aid substantially in improving model performance.

We also experimented with techniques such as SMOTE to generate a training set with balanced labels. However, we observed that while this approach aimed to address class imbalance, it resulted in overfitting.

4.3 BERT Fine-tuning Experiments

We investigate the effectiveness of fine-tuning BERT for our classification task.

4.3.1 Experiment Options

Fine-tuning BERT involves selectively freezing certain layers or parameters of the BERT model, typically lower-level encoder layers, while allowing higher-level layers to be fine-tuned using task-specific labeled data. By immobilizing these lower-level representations, freeze fine-tuning aims to stabilize the model's foundational features while enabling adaptation to task-specific nuances in higher-level layers.

There are different methods of fine-tuning BERT. However, due to the significant computational cost and marginal improvement in results (even less than that achieved by early stopping), we opted to freeze the BERT parameters. Downstream fine-tuning was done by adding fully-connected layers after the BERT layers. The effect of this is equivalent to using BERT as a feature extractor. According to the results of the SciBERT Paper[2](Beltagy et al., 2019), we set the epochs no more than 10. And we use Adam as the optimizer for a more stable and faster optimization process.

4.3.2 Model and Improvement Process

Similar to the LSTM approach, we consider adding sectionName as an additional feature to aid in classification. This feature was concatenated to the input to the last fully connected layer.

Lastly, we investigate the effect of changing the corpora these pre-trained models were trained on. Specifically, we experimented with the SciBERT model, that was specifically trained on scientific text, in contrast to BERT which was pre-trained on books and Wikipedia data.

From our results, we conclude that the transformer architecture itself does not directly imply better results than a more traditional approach like LSTM with Word2Vec. We also observe a significant improvement in performance by using SciBERT as opposed to BERT, as SciBERT benefits from contextual learning from pre-training on scientific text.

Since this form of fine-tuning is more akin to feature extraction, rather than parameter fine-tuning of BERT, we proceed to explore the difference in semantic information captured by Word2Vec embeddings vs BERT embeddings. (Please refer to

[2]https://doi.org/10.48550/arXiv.1903.10676

5.1 for details about embedding selection) Furthermore, we investigate whether LSTM models capture comparable semantic information to BERT embeddings when using Word2Vec embeddings as inputs. (Please refer to 5.2 for details about embedding selection)

4.4 Combine LSTM and Self Attention Mechanism

Macro Precision	Macro Recall	Macro F1	Micro F1
0.84	0.84	0.84	0.86

Table 8: UniLSTM with SciBERT tokenizer and self-attention

Rather than utilizing LSTM and BERT independently, we explore the approach of combining SciBERT with LSTM and self-attention mechanisms. Each component of this approach brings complementary strengths to the model. SciBERT provides domain-specific knowledge, allowing the model to benefit from transfer learning, while the uniLSTM captures local sequential dependencies which proves to be especially important for this classification task. Lastly, self-attention mechanisms enable flexible feature extraction to capture global semantic information. This combination allows the model to better understand the text content at different levels and make classification decisions based on both global and local information. This combination yields our best model with a macro F1-score of 0.84.

5 Discussion

Our experimental results interestingly reveal that the BERT transformer architecture does not necessarily perform better than a traditional LSTM architecture with Word2Vec inputs. In this section, we explore the difference in semantic information captured by Word2Vec embeddings vs BERT embeddings. Furthermore, we investigate whether LSTM models capture comparable semantic information to BERT embeddings when using Word2Vec embeddings as inputs.

5.1 Selection of word embeddings

We experiment with four word embeddings - self-trained Word2Vec, Google pre-trained Word2Vec, BERT pre-trained, SciBERT pre-trained - to represent our input string[3]. We visualize this input string

[3]We visualize this input string - "Methods such as SBDS [2, 18] avoid this problem but adding constraints to a model does not, so measuring the search effort to find a solution would not be a meaningful way of evaluating the new encoding."

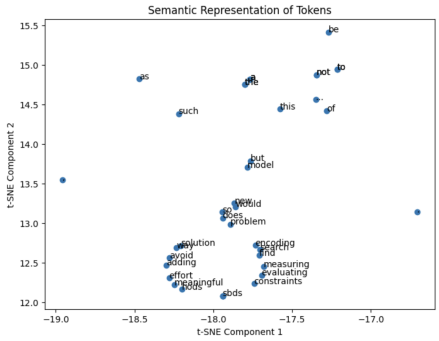

Figure 5: Self-trained Word2Vec Embedding

in Figure 5(Above),13,14,15(In the appendix).

We observe that the Word2Vec embeddings capture significantly lesser semantic relationships between tokens in general. While the self-trained Word2Vec embedding can express general language semantic relationships, the BERT/SciBERT embeddings can capture much richer and context-relevant relationships. The token representations are also much more clustered in the BERT/SciBERT embeddings compared to the Word2Vec embeddings, representing stronger semantic relationships.

The subpar performance of the fine-tuned BERT model, compared to the uniLSTM with Word2Vec model, can hence be attributed to BERT's pre-training on a dissimilar corpus type. This renders transfer learning ineffective, and inferior compared to learning from scratch by the LSTM on the Word2Vec vectors of the SciCite dataset.

5.2 Modelling LSTM with Word2Vec to mimic SciBERT

From the visualisation of the word embeddings, we observe that the SciBERT model acts as a powerful feature extractor in capturing complex context-based semantic relationships, that the Word2Vec embeddings lack. We investigate if different LSTM models are able to mimic BERT's ability to represent these semantic relationships.

5.2.1 Effect of Capturing Bi-directional Dependencies

We analyze the effect of capturing bi-directional dependencies on the performance of our classification task, by replacing the uniLSTM layer with a BiLSTM layer.

The performance of the models are very similar, both when aggregated, and also for individual classes and metrics. This could be because scientific citations often follow a structured format, with clear indications of the cited work and its rele-

Table 9: UniLSTM vs BiLSTM

	Precision	Recall	F1-score
Class 0	0.83	0.90	0.86
Class 1	0.94	0.76	0.84
Class 2	0.72	0.79	0.75
Micro avg			0.84
Macro avg	0.83	0.82	0.82
Weighted avg	0.85	0.84	0.84
Class 0	0.85	0.90	0.87
Class 1	0.94	0.79	0.85
Class 2	0.71	0.80	0.75
Micro avg			0.83
Macro avg	0.83	0.83	0.83
Weighted avg	0.86	0.85	0.85

vance to the current research. Therefore, citation intent can typically be discerned from semantic cues surrounding the citation without requiring bidirectional dependencies.

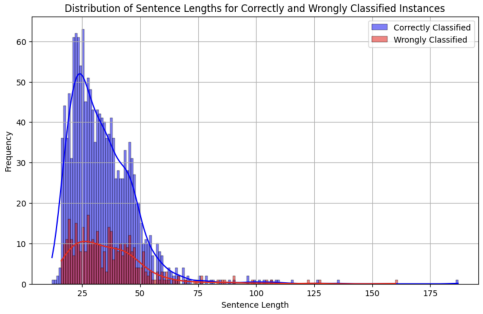

Figure 6: Distribution of Sentence Lengths for Correctly and Wrongly Classified Instances for UniLSTM

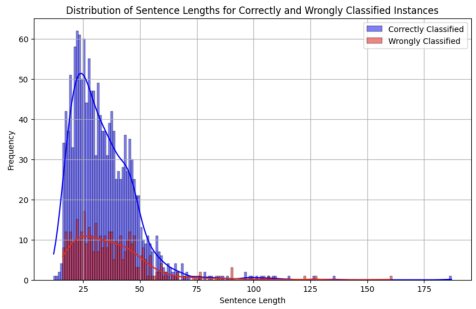

Figure 7: Distribution of Sentence Lengths for Correctly and Wrongly Classified Instances for BiLSTM

There is a negligible difference in the distribution of sentence length in correctly and wrongly classified instances across both architectures. First, we note that the Word2Vec representation of our input sentences are truncated at 33 tokens. That means that we effectively ignore parts of longer sentences that exceed the 33 token window size. From our results, we can conclude that the uniLSTM does as well as the biLSTM in capturing long-range dependencies within this 33 token window size. Therefore, for our setup, it is futile to swap out our uniLSTM layer for a biLSTM layer to process longer input strings. However, further experiments can be conducted to investigate if the biLSTM model performs better as the window size increases.

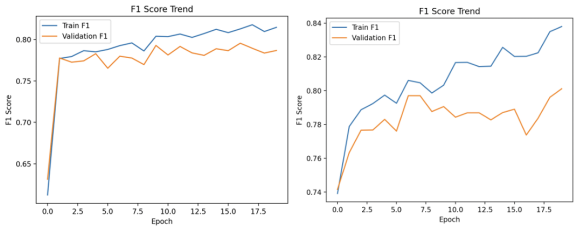

Figure 8: UniLSTM vs BiLSTM Training Process

The training process reveals significant overfitting for the biLSTM model, compared to the uniLSTM model. Due to its bidirectional nature, the biLSTM model inherently has a higher capacity to capture and model complex dependencies within the data. This higher capacity makes the biLSTM model more prone to overfitting.

In fact, we observe that the biLSTM layer can misrepresent some relationships between tokens[4]. For example, 'adding' shows strong semantic relationship to 'SBDS' by the BiLSTM hidden states, while 'SBDS' shows stronger semantic relationship with 'methods' instead in the UniLSTM hidden states. This difference in semantic representation could be attributed to the biLSTM's ability to capture bidirectional dependencies. However, the representation by the UniLSTM hidden states is more accurate and also leads to an accurate classification of the instance.

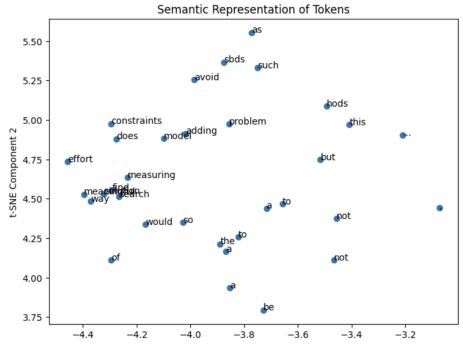

Figure 9: UniLSTM Hidden State

5.2.2 Effect of Adding Self-Attention

We analyze the effect of incorporating a self-attention mechanism in our LSTM architecture on the performance of our classification task. This subset of experiments were conducted on a different deep learning library as the others, but variables were controlled to ensure the validity of the experimental results.

[4]The sentence in question is "Methods such as SBDS [2, 18] avoid this problem but adding constraints to a model does not, so measuring the search effort to find a solution would not be a meaningful way of evaluating the new encoding."

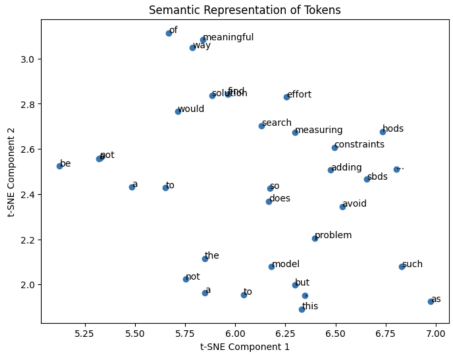

Figure 10: BiLSTM Hidden State

The LSTM output contains the hidden states from each time step of the input sequence. The attention mechanism is applied to the LSTM output to calculate attention scores. The attention weights represent the importance of each time step's hidden state in the context of the entire sequence. We then feed in the weighted LSTM outputs into a fully connected layer.

Table 10: UniLSTM vs UniLSTM with Attention trained on Google Word2Vec

	Precision	Recall	F1-score
Class 0	0.86	0.72	0.78
Class 1	0.81	0.86	0.84
Class 2	0.65	0.76	0.70
Micro avg			0.80
Macro avg	0.78	0.78	0.77
Weighted avg	0.81	0.80	0.80
Class 0	0.86	0.75	0.80
Class 1	0.82	0.87	0.84
Class 2	0.70	0.74	0.72
Micro avg			0.81
Macro avg	0.79	0.79	0.79
Weighted avg	0.81	0.81	0.81

We observe an all-around improvement in performance metrics, albeit a small dip in recall of class 2. Macro and micro, as well as individual classes' F1-scores all improved.

The distributions in sentence length of correctly classified and wrongly classified instances are very similar for both models. This suggests that the self-attention mechanism does not improve the classification of longer sentences. This could be because the Word2Vec representation of our input sentences are truncated at 33 tokens and the uniLSTM layer itself can capture the dependencies within this 33-token window well enough. More experiments can be conducted to investigate if self-attention aids in capturing long-range dependencies for window sizes over 33.

Focusing on the training process, we observe that the loss converges to the minimum of around 0.5 much earlier, at around the 4th epoch for the model with self-attention, as compared to the 8th

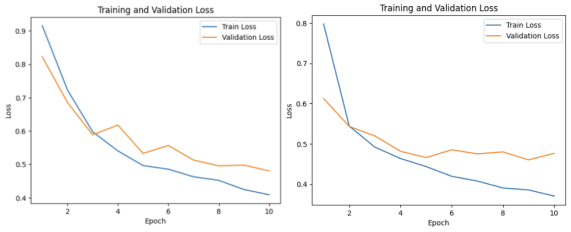

Figure 11: UniLSTM vs UniLSTM with Self-Attention Training Process

epoch in the model without. The self-attention mechanism helps to weight the hidden states of the LSTM output beforehand, helping the model to learn the weights faster and reducing the gradient descent needed to optimize the model.

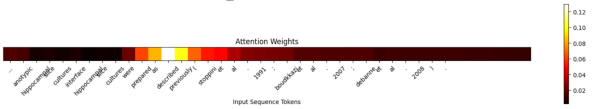

Figure 12: Attention Weights

Given this sentence, "Organotypic hippocampal slice cultures Interface hippocampal slice cultures were prepared as described previously (Stoppini et al. 1991; Boudkkazi et al. 2007; Debanne et al. 2008).", the self-attention mechanism weights the hidden states as given in the figure above. The highest weights were given to tokens such as "prepared as described previously", which aids the model in understanding and classifying this citation instance as a method.

6 Conclusion

Our study utilized traditional machine learning techniques and advanced models such as LSTM and BERT to improve the classification of academic documents. The best results were obtained by using SciBERT embeddings, fed into an LSTM model with a self-attention mechanism, achieving a notable macro F1 score of 0.84 and micro F1 score of 0.86. However, the study had limitations. The fine-tuning of BERT was restricted to freezing certain layers; more intricate fine-tuning strategies might yield better results. Additionally, the fixed input sequence length of 33 for LSTM, based on average text length, could be reassessed for a more scientific approach. Looking forward, exploring these avenues could potentially enhance the model's accuracy and adaptability, broadening its applicability to more diverse datasets.

References

Iz Beltagy, Kyle Lo, and Arman Cohan. 2019. Scibert: Pretrained language model for scientific text. In *EMNLP*.

Arman Cohan, Waleed Ammar, Madeleine Van Zuylen, and Field Cady. 2019. Structural scaffolds for citation intent classification in scientific publications. *arXiv preprint arXiv:1904.01608*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding.

Shengfeng Gan, Shiqi Shao, Long Chen, Liangjun Yu, and Liangxiao Jiang. 2021. Adapting hidden naive bayes for text classification. *Mathematics*, 9(19).

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Thorsten Joachims. 1998. Text categorization with support vector machines: Learning with many relevant features. In *European conference on machine learning*, pages 137–142. Springer.

Mohammad Sadegh Rasooli and Joel R. Tetreault. 2015. Yara parser: A fast and accurate dependency parser. *Computing Research Repository*, arXiv:1503.06733. Version 2.

Kanish Shah, Henil Patel, Devanshi Sanghvi, and Manan Shah. 2020. A comparative analysis of logistic regression, random forest and knn models for the text classification. *Augmented Human Research*, 5(1):12.

Statement of Independent Work

You **must** *include the text of the two statements below in your group's submitted work. Digitally sign your submission using your Student Numbers (starting with A...; N.B., not your NUSNET email identifier). This is a required section and is not part of the main body (doesn't count towards your page limit).*

1A. Declaration of Original Work. By entering our Student IDs below, we certify that we completed our assignment independently of all others (except where sanctioned during in-class sessions), obeying the class policy outlined in the introductory lecture. In particular, we are allowed to discuss the problems and solutions in this assignment, but have waited at least 30 minutes by doing other activities unrelated to class before attempting to complete or modify our answers as per the class policy.

We have documented our use of AI tools (if applicable) in a following table, as suggested in the NUS AI Tools policy[5]. This particular document did not use any AI Tools to proofcheck and was constructed and edited purely by manual work.

Signed,
A0259561W, e0970533@u.nus.edu
A0275766L, e1127420@u.nus.edu
A0238146A, e0772744@u.nus.edu
A0275725W, e1127379@u.nus.edu
A0253860A, e0952398@u.nus.edu

[5] https://libguides.nus.edu.sg/new2nus/acadintegrity, tab "AI Tools: Guidelines on Use in Academic Work"

A Appendix

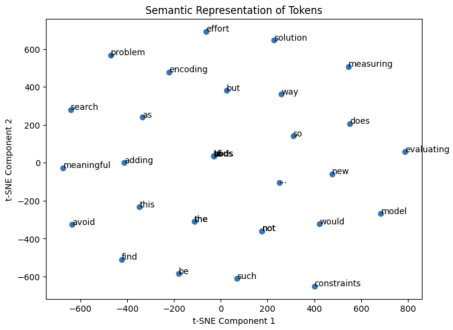

Figure 13: Google Word2Vec Embedding

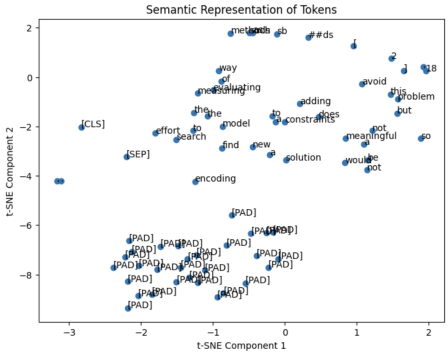

Figure 14: BERT Embedding

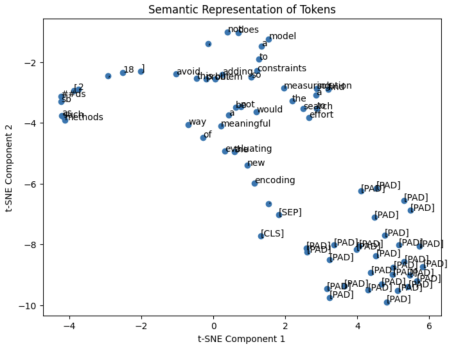

Figure 15: SciBERT Embedding

Name	Description
source	The source of the citation. (String)
sectionName	The name of the section the citation is found in. (String)
string	The string of text associated with the citation. (String)
label	The label associated with the citation. (String)
label2	The second label associated with the citation if label is “result”. (String)
labelConfidence	Confidence of label for string. (Integer)
label2Confidence	Confidence of label2 for string. (Integer)
isKeyCitation	The indicator whether the string is a key citation used by others. (Boolean)
citeStart	The start index of the citation in the text. (Integer)
citeEnd	The end index of the citation in the text. (Integer)
citingPaperId, citedPaperId, id, uniqueId, exceptIndex	Ids associated. (String / Integer)

Table 11: Dataset