CS4248 Final Report

A0201815R, A0219668M, A0220741U, A0222371R, A0248400N, A0275768H
Group 14
Mentored by Liu Chenyan
{e0415624,e0550326,e0555933,e0559903,e0925552,e1127422}@u.nus.edu

Abstract

In this paper, we explore classifying citation intent in research papers (*background, methods, results*), by applying different architectures and models. Specifically, we focus on three areas: how the models compare on a high level, how specific methods can be used to model task complexity more robustly, and which features best capture semantic information that is most helpful for this task. The RNN variants performs best, but with interesting observations about how they interact with the attention mechanism. In particular, we found a nuanced relationship between model complexity and performance beyond the bias-variance tradeoff, while also observing some subtleties about certain features, such as the differential effects of word and word order on predictions. Ultimately, we hope to demonstrate that interesting insights can be derived on these comparatively simple models.

1 Introduction

Citations are a key part of scientific research, as they provide corroboration for the point being made, offer insights on prior approaches, indicate prior results as a point of comparison, and provide a basis on which improvements can be made. When and how a particular research publication is cited plays an important role in judging the scientific significance of the publication as well as possibly validating its quality (Aksnes et al., 2019). This can then be leveraged for different purposes – improving information retrieval on scientific papers (Xu et al., 2013; Bertin and Atanassova, 2012), understanding the evolution and progress of a scientific field (Hassan et al., 2017) as well as citation-based summarisation (Jha et al., 2017; Fisas et al., 2016).

To this end, automated citation intent classification is a common research problem that aims to provide clarity on citations *en masse*. By classifying citations, for example by sentiment or by their role in the text, we can quantify the impact of that citation. This is most often done by examining the words or sentences that immediately surround the citation (Garzone and Mercer, 2000; Jurgens et al., 2018), also known as the citation context. Output classes, however, tended to vary between different studies. For instance, Zhu et al. (2015) categorised citations into "Influential" / "Noninfluential", while Lauscher et al. (2017) classified them based on both polarity and purpose. Broadly speaking, these studies classified citations based on their function (e.g. "comparing", "contradicting", "background information"), importance, and polarity (e.g. "positive", "negative") (Kunnath et al., 2021).

In our work, we focus on classifying citations based on their surrounding context, obtained from papers in computer science and medicine domains (Cohan et al., 2019). Each piece of input typically includes one to two sentences which surround one or more citations, and we classify them by their function, into *background*, *method* or *result*. We explore several different hypotheses regarding the choice of model, specific architecture of the model, as well as preprocessing and feature engineering. First, we investigate the effectiveness of increasing complexity in the model, as well as performance of deep learning models as compared to traditional machine learning models. Next, the impact of different model components, including attention. Lastly, the impact of word order and static features, such as the section name and position of citation context within the passage, with regards to the different models.

2 Related work / Background

Research into automated citation classification schemes first began with Garzone and Mercer (2000) and their rule-based scheme, consisting of lexical matching and parsing rules. Machine Learning (ML) models, however, quickly took over after the 2010's (Kunnath et al., 2021), with researchers

first applying traditional ML models such as Support Vector Machines (Xu et al., 2013; Kim and Thoma, 2015) and Naive Bayes (NB) (Widyantoro and Amin, 2014). The focus then transitioned to deep learning (DL) methodologies such as Long-Short-Term Memory (LSTM) with attention (Munkhdalai et al., 2016), Convolutional and Recurrent Neural Networks (CNN/RNN) (Yousif et al., 2019), and pre-trained transformers (Maheshwari et al., 2021).

Most of the work done on citation function classification either focused on comparison of a cutting-edge DL model with a baseline classical ML model or comparing a few different classical ML models. For instance, Munkhdalai et al. (2016) compared their novel attention-augmented LSTM model with a baseline SVM model using TF-IDF and n-grams. Similarly, Aljohani et al. (2021) applied a CNN architecture and compared it to SVM and NB with TF-IDF. Abu-Jbara et al. (2013), meanwhile, compared the performance of SVM, Logistic Regression (LR) and NB. In addition, not much research has been done on the Multi-Layer Perceptron (MLP) model. Maheshwari et al. (2021) utilised a transformer connected to a single linear layer, however their work was more focused on the effectiveness of different transformer architectures. Mishra and Mishra (2020) achieved the best results with MLP as compared to LR and Random Forest, suggesting that MLP was an under-researched area that could potentially yield good results. This was further compounded by the lack of availability of a widely-used, large-scale dataset, meaning that different authors used different datasets which made meta-analysis difficult.

As such, one focal area of our work is to assess the performance of different DL models, as well as MLP and Logistic Regression. We operate on the SciCite dataset (Cohan et al., 2019), which should result in our results being more generalisable as compared to prior works that operated on much smaller datasets. Rather than focusing on novel architectures or state-of-the-art designs, our aim is to study the relative performance of different models on the same dataset and investigate the potential of different architectures.

In addition, we aim to investigate how different architectures for the same model could better model the task complexity, for instance how the presence of an attention layer affected the results. Munkhdalai et al. (2016), found that adding attention to Bi-LSTM resulted in the best performance, but was only a slight improvement over vanilla Bi-LSTMs; Cohan et al. (2019) used LSTM and Bi-LSTM with attention as a baseline, but did not explore designs without attention.

Lastly, prior work that applied traditional ML models used manually engineered features such as POS tags (Jochim and Schütze, 2012), dependency relations (Jochim and Schütze, 2012), and location of citation (Jurgens et al., 2018); while DL-based research typically did not apply much manual feature engineering, instead opting for word embeddings such as Word2Vec (Munkhdalai et al., 2016), fastText (Aljohani et al., 2021) and BERT (Maheshwari et al., 2021). Hence, rather than repeating their experiments regarding feature engineering, we focus on investigating the impact of manually engineered features, both contextual and non-contextual, on DL architectures.

3 Corpus Analysis & Method

3.1 Dataset

There are multiple datasets of citations found in scientific literature that contain intent annotations. The ACL-ARC citations dataset (Jurgens et al., 2018) contains 1,941 citations in the field of Computational Linguistics, classified into 6 citation intent categories. Meanwhile, the SciCite dataset is significantly larger, containing 11,020 citation instances from scientific papers in the Computer Science and Medical domains. These citations are classified into 3 categories: Background, Method and Result Comparison. The dataset consists of the sentence containing the citation and the associated label, as well as a few contextual and non-contextual features; it is also unbalanced across the categories. In this paper, we utilize the SciCite dataset due to its significantly larger size.

3.2 Preprocessing

The SciCite dataset contains the sentence containing the citation and its annotated intent. The input sentences were first tokenized. In Logistic Regression and MLP models, stop word removal was performed followed by feature extraction. For RNN and LSTM, the input feature at each timestep was a 200-dimensional GloVe word embedding (Jeffrey Pennington and Manning, 2014).[1]

[1]Our analysis code can be found at: https://github.com/justinfidelis/CS4248-Project

3.3 Evaluation of performance of different model architectures

First, we evaluate the comparative performance of different models and architectures to gain deeper insight into their strengths and limitations. We train and measure the performance of the following model types: Multinomial Logistic Regression, MLP, RNN, and LSTM.

Evaluation of Logistic Regression and MLP. In classification tasks, the relationship between input features and the correct label is oftentimes not immediately obvious. Theoretically, models with increased complexity should be able to better learn these complex functions and achieve reduced bias. However, model variance also increases with complexity, increasing the likelihood of overfitting.

We trained and tested the performance of multiple MLP models with different numbers of nodes and layers. A multinomial logistic regression model was used as an additional baseline as it is equivalent to an MLP with no hidden layers, and which uses a softmax function on its output. To evaluate the model architectures' ability at learning different functions, different combinations of input features were used.

Comparison of LSTM against baseline RNN. One of the main limitations of RNNs is its poor performance when working with longer input sequences due to the vanishing gradient problem (Hochreiter and Schmidhuber, 1997). LSTM models address this problem with the addition of a more complex memory cell architecture better suited to storing information over a larger number of time steps. Bi-LSTM, meanwhile, transmits information across the sequence in both forward and backward directions, allowing for dependencies in both directions to be captured. We compared the performance of these three architectures.

3.4 Study of methods to model task complexity

Next, we identify specific model components that could be essential for the modeling of nonlinear features and relationships. This is motivated by the intuition that when trying to evaluate citation intent, we instinctively reconcile many non-obvious contextual and linguistic cues. We aim to shed some light on the inherent complexity of the task, as well as the functional nuances of these components.

Value of attention layer in encoding complex semantic information. We hypothesized that a vanilla RNN/LSTM fails to capture a lot of semantic information encoded non-sequentially within a sentence. For example, the following *background* sentence *"whereas indirect benefits refer to benefits to genetically related recipients; i.e., benefits that increase the inclusive fitness of benefactors but typically imply fitness costs that are not compensated during the benefactors' lifetimes"* serves to provide context to the concept of 'indirect benefits'. As the anchor concept in the sentence, 'indirect benefits' is most directly explained by the phrases 'inclusive fitness' and 'not compensated', but these specific associations may be lost in the naive feed-forward logic of a vanilla RNN. As such, we think that an attention layer should substantially improve the predictive power of most variants of RNNs.

Impact of additional hidden layers and activation functions. A point of interest for us is the complexity of the task. Intuitively, one approach to unpack this is to explore the nature of the relationship between the static features - e.g. positional indices of the section, section name, etc. - and the word embeddings, in relation to the prediction task. For example, we can speculate that introductory paragraphs might include more critical descriptions of existing literature, thus a context vector representing text of such a nature, combined with indices suggesting an earlier position in the paper, might be more indicative of a 'background' citation type. We test the depth of these complexities by introducing a hidden layer to the combined component of the network, where the static features have been conjoined with the attention context. Furthermore, we compare this to a variant where the additional hidden layer is instead attached to the attention layer, essentially comparing the degree of nonlinearity inherent to the contributions of semantic information with that of the interaction effects between semantic and static features.

3.5 Choice of features to capture data semantics

For this question, we explored the significance of various methods of feature generation and engineering in the context of RNN models. In doing so, we aim to uncover axes of considerations that we can use that yield good features that better capture the semantics of the data.

Impact of word order on prediction. Within the RNN family of models, we considered the effects of randomizing the sequence of input words. Specifically, we experimented with two variants of the RNN model, namely randomization of words across the entire input sequence, and randomization of n-grams of words, where the order of words in each n-gram is maintained as in the original input sequence. This serves to investigate if an RNN equipped with an attention mechanism can still be effective when word order information is progressively removed. We hypothesized that a greater degree of randomization will diminish the RNN model performance, due to the loss of contextual information in sequences of words when such sequences are randomized. We tested this with a basic RNN model with static features and an attention layer.

Contribution of static features. The second and more trivial hypothesis related to this question is that the given features encode valuable information not captured by the word embeddings. For example, section name, the position of the cited sentence in the passage and isKeyCitation are important features that capture contextual information aside from the text itself. We test this hypothesis by comparing the results of our RNN model with and without these static features.

Furthermore, we observe two types of section names - those that directly feature the source label, e.g. 'Materials and methods' and 'Results and discussion', and those that do not (see Appendix 2). We hypothesize that instead of improving predictions, the latter group may instead introduce some unwanted noise to the model. We test this by attaching the corresponding section names to the front of each string, before the <bos> tag.

4 Experiments

4.1 Evaluation of performance of different model architectures

For this analysis, we used the 200-dimensional GloVe embeddings for each token as the input feature. For Logistic Regression and MLP, the mean of the embedding vectors in each sentence was used due to the inability of those models to take in sequential data, resulting in a "bag-of-words" representation. For RNN and LSTM, the vectors were passed to the model in sequence. F1 score was used as the performance metric.

Model	Test F1 Score
Logistic Regression	0.711
MLP	0.723
RNN	0.435
LSTM	0.753

Table 1: Baseline performance of different models

From the results, we observe that Logistic Regression and MLP perform similarly, with MLP having slightly higher performance.

We also observe that RNN has a significantly lower performance compared to the first 2 models and we hypothesize that this is due to RNN's limitation when working on longer sequences. This hypothesis is somewhat corroborated by the increase in performance with the LSTM model that is better able to store information across longer sequences. LSTM also outperforms Logistic Regression and MLP, likely due to its ability to take word order into consideration.

Evaluation of Logistic Regression and MLP. To investigate the impact of model complexity on MLP performance, different numbers of hidden layers were used. The layer sizes for each feature combination and the hyperparameter settings are provided in Appendix 1A. Additionally, we use logistic regression model as an additional linear baseline model.

Three separate feature sets were used: POS tag counts, word count vector, and GloVe word embeddings (see Appendix 1B for details). To account for variations in model performance due to its random initialisation and stochastic training, 5 models were trained for each combination and the average F1 score was computed.

F1 Score	LR	MLP		
		0 HL	1 HL	2 HL
POS Tags	0.387	0.390	0.395	0.398
Word Count	0.717	0.710	0.712	0.708
Embedding	0.717	0.716	0.721	0.724
All Features	0.766	0.759	0.752	0.749

Table 2: Performance of Logistic Regression (LR) and MLP with different number of Hidden Layers (HL)

From the results in Table 2, there appears to be a slight improvement in model performance for the POS tag and Word embedding features as the MLP complexity increases but a slight reduction in

performance when the word count features and all three features were used. The changes in model performance for each of the three features were small, which could be due to the relationship between the input and output being near-linear in nature, and the addition of non-linearity in MLP providing little additional capability in improving the learnt function.

For the feature combinations with a reduction in model performance, we hypothesize that it was due to increased overfitting as the model complexity increased. This is corroborated by an observed decrease in performance as the number of epochs increases. A plot of this can be found in Appendix 1C.

Comparison of LSTM against baseline RNN. We tested the performance of vanilla RNN against LSTM and bi-directional LSTM; architecture and hyperparameters of the models are in Appendix 1D. GloVe word embeddings were used as the input value (Appendix 1B).

Model	Test Data	
	Accuracy	F1-Score
RNN	0.452	0.435
LSTM	0.782	0.762
BiLSTM	0.795	0.762

Table 3: Comparison of Log Reg and MLP of different complexities

From our results, we observe a significant increase in performance from RNN to LSTM, in line with our expectation due to LSTM's ability to handle long sequences. Additionally Bi-LSTM had a slight improvement in performance over LSTM, which can also be attributed to its ability to capture dependencies in both directions.

4.2 Study of methods to model task complexity

Value of attention layer in encoding complex semantic information

1. RNN

Table 4 shows the scores of the RNN performance with and without the attention layer.

From the results, it is clear that the attention layer is crucial to predictive performance for RNN, allowing it to outperform LSTM and Bi-LSTM. In the example sentence below, where the source has been labeled as 'method', the attention vector

Variation	Acc	F1
Vanilla RNN	0.334	0.435
RNN w Attention	0.756	0.803

Table 4: Vanilla RNN (control)

correctly assigns greater weight to the words 'collected' and 'data', clearly highlighting the value of the attention mechanism in evaluating complex semantic relationships. For this sample, as seen in Figure 4 (Appendix 3) the model predicts the source label correctly.

On the other hand, as seen from the negative sample in Figure 5 (Appendix 3). The model wrongly predicts *background*, and overweighs words such as 'phase' and 'however' that are more ambiguous as intent signals, while missing out the more critical phrase that is 'clinical trial'. Indeed, this sample is ambiguous even to a human eye, suggesting perhaps that the model might struggle where source intent is less clear from semantic or lexical cues alone.

2. LSTM

We observe similar results for the LSTM model. Interestingly, the effect of the self-attention layer seems to be more pronounced for LSTM than it is for Bi-LSTM. In terms of numbers, BiLSTM with the self-attention layer and a dropout of 0.5 performs the best, achieving a test accuracy and macro f1-score of 0.805 and 0.775 respectively. These performance scores can be viewed in Table 13.(Appendix 3)

Impact of additional hidden layers and activation functions Here, we start with a baseline RNN model with attention, static features and sections headers (see Table 15 and 16 in Appendix 3); the static features and section header will be introduced in Section 4.3 below. This was then modified with an additional hidden layer 1) at the combined section and 2) at the attention layer. Embeddings, static features and section names were used as input features.

Variation	Acc	F1
RNN Baseline	0.849	0.880
RNN augment at Combined	0.853	0.879
RNN augment at Attention	0.844	0.878

Table 5: RNN performance

From the above, we observe that adding a hidden layer seems to have negligible impact on performance, regardless of its location. This might suggest either that the amount of non-linearity in the relationships between embeddings and static features has been quite exhaustively captured by the baseline structure, or that a hidden layer is unable to capture that relationship. Given the limited complexity that can be be present in a single sentence, the former is indeed plausible. This intuition is further corroborated by the results obtained from the SwiGLU and Swish tests on the RNN model, which have been omitted from the body text for brevity (Appendix 2).

4.3 Choice of features to capture data semantics

Impact of word order in prediction. To investigate the effects of word order on the accuracy of the RNN, we added an additional preprocessing step to the corpus to either: randomize the order of all words in each sentence, or to split each sentence to n-grams with given n, and randomize across n-grams. Again, we start with a baseline of RNN with attention, static features and section headers.

We performed randomization with different values of n, and compared the RNN's performance. Additionally, a dynamic setting of n, with $n = sentenceLength/5$, was also explored

Modification	F1 Score
No Randomization	0.880
Full Randomization (n=1)	0.854
Randomization (n=5)	0.860
Randomization (n=10)	0.880
Randomization (dynamic n)	0.868

Table 6: Impact of word order randomization on model performance

We observe that the control consistently outperformed all three modifications. Across the modifications, we see that a greater degree of randomization resulted in a lower performance. That said, it is surprising that the impact of randomization on f1-score is relatively minor. This suggests two conclusions: firstly, that the word order contains some information relevant to the classification task that can be exploited by the model, and secondly, that in the absence of sequenced data, a bag-of-words representation of input data is a poorer but still viable substitute.

Contribution of static features

1. Aggregate impact of static features

The performance of the RNN model with attention only was evaluated with word embeddings alone and with both embeddings and static features.

Modification	F1 Score
No Static Features	0.803
Static Features	0.847

Table 7: Impact of static features in RNN performance

From the results, it is clear that the model performs significantly better with static features. Embedding information seems to encode subtler signals, while we also observe that 'label 2' and 'isKeyCitation', which correspond to the 5 columns on the right, seem to have a strong effect on predictions. Specifically, the labels 'supportive' and 'not supportive' correlate distinctly to the 'result' intent, which makes sense as they indicate the text's position on the paper's hypothesis.

2. Impact of section name

Table 16 in Appendix 3 shows the difference in the results obtained when section names are included. The baseline model used is the RNN with attention and static features.

As conjectured, the inclusion of section names significantly boosts model performance across all classes. We also try to isolate the pure effect of section names by evaluating the model after excluding section names that directly feature the citation labels. Indeed, we observe that this distilled effect is more nuanced. While precision and recall are marginally higher, balanced accuracy is slightly lower, indicating that the model's specificity is actually lower. Based on the classification report for the validation dataset, a possible explanation could be the larger false positive rate for a non-majority class, although these differences vis-a-vis the control case are too small for a significant conclusion to be made.

In the examples of the attention vector in Figure 6 and Figure 7 Appendix 3, we observe how the 'methods' section name was proportionally over-weighted before its exclusion, after which the weight allocations more closely resembled those obtained earlier in the findings for hypothesis 2, although the latter is not always the case. A surprising observation is that the <bos> tag is often

given more emphasis than the rest of the text. We suspect that this is due in part to its role as a critical juncture in the sequence, which causes it to have a substantial relationship with both the section name and the cited text. Finally, we note that there are many cases where 'direct' section names do not correspond to source intent, leading to misleading signals that do not enhance predictions.

5 Discussion

5.1 Evaluation of performance of different model architectures

Evaluation of Logistic Regression and MLP. To gain some insight into the functions that were learnt by the two model types, we analyzed them using feature attribution. A logistic regression model and three MLPs with different numbers of hidden layers were trained on the token count vector. The learnt weights of the logistic regression model is a direct measure of the relative importance of each token in classification. For MLP, the integrated gradients method (Mukund Sundararajan and Yan, 2017) as implemented by the captum package was used to calculate the attribution scores for each token.

For example, we observe the *results* sentence *"Our findings agree with recent work [69,70] where continual organic enrichment from farming processes resulted in increased macrofaunal abundances despite expectations of negative impacts from this contamination."*. The words with the highest contribution to the *results* label are very similar across all models, with the models having the same top 5 highest contributing words: *findings, agree, despite, resulted, recent*. The contribution of these words makes sense as these words are commonly used when comparing research findings.

From the result, we can conclude significant convergence in the learnt functions across the models, even despite the differences in architecture. A plot of the full set of contribution scores can be found in Appendix 1E.

5.2 Study of methods to model task complexity

Overall, we have gleaned a nuanced view of how the tested model components relate to the citation intent problem. Beyond the overfitting concern, one should also consider the type of complexity best suited to the core model and data before introducing more of it.

Value of attention layer in encoding complex semantic information Across all the RNN and LSTM variants tested, we observe that the attention mechanism improves model performance quite substantially, albeit with nuances. While its benefit in encoding arbitrary relationships between words is clear, its value to a prediction task is limited by the amount of relevant information encoded by semantic or lexical cues. We see this clearly from how the model may misclassify ambiguous sentences that are harder to classify without context, as illustrated earlier. If the text itself is not helpful, neither is attention. This is most apparent from the drop in accuracy for the method class when applying self-attention to Bi-LSTM. As another example, the *method* sentence *"Clinical reports of energetic stress in HCM hearts (14, 15) prompted us to measure mitochondrial respiration, mitochondrial copy number, ROS emission/scavenging, and calcium handling in these 2 HCM mouse models."* seems like it should be *background* even to a human reader.

Furthermore, we also notice that the attention layer can be quite sensitive to the structure and coherence of its inputs. In the Bi-LSTM case, where the input to the attention layer is a concatenation of two opposing sequences of hidden states, we speculate that the layer might have encoded noisy and incoherent relationships between words that do not make linguistic sense, perhaps contributing to its weaker improvements vis-a-vis the LSTM case. Similarly, we noticed that the inclusion of irrelevant section names seems to throw off the attention mechanism, leading it to attribute substantial value to the header and altering probability rankings. These observations reveal what seems like a sensitivity to miscues and incoherence in the input sequence, suggesting again that the attention mechanism cannot be blindly used, and requires due consideration of the characteristics of the core model and input data.

Impact of additional hidden layers and activation functions We observe quite consistently that adding hidden layers seems to slightly worsen overfitting. This applies to the use of SwiGLU and Swish as well, which supports our conjecture that few meaningful nonlinear features can still be derived from the existing feature set through affine transformations alone, short of adding new features to this baseline set. While one might argue that the observed superiority of SwiGLU to Swish in this

case might imply that some features are redundant or weaker, the negative effect of SwiGLU itself on the control model does not support this conjecture.

An especially interesting result was that the hidden layer yields poorer outcomes when added only to the attention portion. While it is true that a sentence does not encode nearly as much information as a paragraph/article, we note here that the implications of an affine transformation are very different from that of an attention layer. Conceptually, one could see how similarity between hidden dimensions of the previous layer, as encoded by dot products, might reflect the natural characteristics of language better than a linear combination of these same hidden dimensions. For example, when we relate a predicate nominative or an adjective to a subject, we might think of their relationships as more associative than additive. As such, one possible intervention here, in lieu of what we have tried, is to add another attention layer instead of a new linear layer, to give further context to the words or phrases that the first attention layer has already contextualized.

5.3 Choice of features to capture data semantics

Impact of word order in prediction We observed that slight randomization of word order in the input sentence resulted in only a small reduction in performance. We theorize that this is due to the comparatively smaller amount of information encoded in word order as compared to the other features used.

Contribution of static features We observe that the effect of static features on the performance of the RNN model is significant, and hence its impact on performance depends on the quality of the static features in representing the nature of the sentence. This suggests that when running citation classification on entire papers as input (and not just the citation contexts), extracting these features will be highly beneficial.

6 Conclusion

In our results, RNN augmented with attention, static features and section headers performed the best. LSTM and Bi-LSTM did not outperform it, perhaps because the dependencies in the given citation contexts are relatively short-term, or because the added complexity of LSTMs captured more noise or introduced greater overfitting. A common theme in our findings is that the addition of hidden linear layers does not seem to improve model performance, be it when comparing a deeper MLP to a logistic regression model, or when comparing two variants of a RNN. This, coupled with the significant improvements brought by attention, leads us to believe that the issue was not entirely about overfitting, but also about the type of complexity introduced. In the context of analyzing citation intent, which might even be a trickier task than sentiment analysis when we consider the broader lexicon of academic writing and the relative ambiguity between intent types, the additive nature of a hidden layer seems to be less equipped for encoding semantic cues than the more associative nature of an attention layer. That said, as we have shown earlier, attention is no magic pill on its own due to its sensitivity to the structure and characteristics of its inputs.

On the features end, two other interesting results were the weaker importance of POS tags and word order, which may not be unrelated. A possible hypothesis is that the presence of related ngrams/key phrases (e.g. 'descriptive data' and 'collected during observation') is a key signal of citation intent, whereas sentence structures and word orders are far noisier variables as the former signal can be expressed by a wide range of sentence structures. A deeper dive into this hypothesis could be a scope for future research.

Limitations and future directions In our study, we focused on simpler, single-model architectures (e.g. LSTM or MLP as standalone classifiers) instead of combining different models or using more complex, state-of-the-art models. Also, we only focused on classifying citation objectives into *background, method, results*, when citation classification can involve classifying sentiment ("positive", "negative"), importance as well as other types of citation functions. Lastly, we only applied a limited set of feature engineering techniques.

Future work could include architectures combining several models, using sentence embeddings from transformers such as SciBERT. We could also expand our range of feature engineering techniques, such as by grouping sentences with similar dependency parse trees, modifying word embeddings to include POS as well as positional information to verify the hypothesis mentioned in the conclusion, and including parts of the preceding and subsequent texts from the actual papers for greater context.

References

A. Abu-Jbara, J. Ezra, and D. Radev. 2013. Purpose and polarity of citation: Towards nlp-based bibliometrics.

D. W. Aksnes, Liv Langfeldt, and Paul Wouters. 2019. Citations, citation indicators, and research quality: An overview of basic concepts and theories.

N. R. Aljohani, A. Fayoumi, and S.-U. Hassan. 2021. *A novel focal-loss and class-weight-aware convolutional neural network for the classification of in-text citations.*

M. Bertin and I. Atanassova. 2012. Semantic enrichment of scientific publications and metadata: Citation analysis through contextual and cognitive analysis.

A. Cohan, W. Ammar, M. van Zuylen, and F. Cady. 2019. *Structural scaffolds for citation intent classification in scientific publications.*

B. Fisas, F. Ronzano, and H. Saggion. 2016. *A multi-layered annotated corpus of scientific papers.*

M. Garzone and R. E. Mercer. 2000. *Towards an automated citation classifier.*

S.-U. Hassan, A. Akram, and P. Haddawy. 2017. *Identifying important citations using contextual information from full text.*

S. Hochreiter and J. Schmidhuber. 1997. *Long short-term memory.*

Richard Socher Jeffrey Pennington and Christopher D. Manning. 2014. *GloVe: Global Vectors for Word Representation*.

R. Jha, A.-A. Jbara, V. Qazvinian, and D. R. Radev. 2017. *NLP-driven citation analysis for scientometrics.*

C. Jochim and H. Schütze. 2012. *Towards a generic and flexible citation classifier based on a faceted classification scheme.*

D. Jurgens, S. Kumar, R. Hoover, D. McFarland, and D. Jurafsky. 2018. *Measuring the evolution of a scientific field through citation frames.*

I. C. Kim and G. R. Thoma. 2015. *Automated classification of author's sentiments in citation using machine learning techniques: A preliminary study.*

S. N. Kunnath, D. Herrmannova, D. Pride, and P. Knoth. 2021. *A meta-analysis of semantic classification of citations.*

A. Lauscher, G. Glavaš, S. P. Ponzetto, and K. Eckert. 2017. *Investigating convolutional networks and domain-specific embeddings for semantic classification of citations.*

H. Maheshwari, B. Singh, and V. Varma. 2021. *Scibert sentence representation for citation context classification.*

S. Mishra and S. Mishra. 2020. *Scubed at 3C task A - A simple baseline for citation context purpose classification*.

Ankur Taly Mukund Sundararajan and Qiqi Yan. 2017. Axiomatic attribution for deep networks. In *Proceedings of the 34th International Conference on Machine Learning*.

T. Munkhdalai, J. P. Lalor, and H. Yu. 2016. *Citation analysis with neural attention models.*

D. H. Widyantoro and I. Amin. 2014. *Citation sentence identification and classification for related work summarization.*

H. Xu, E. Martin, and A. Mahidadia. 2013. *Using heterogeneous features for scientific citation classification.*

A. Yousif, Z. Niu, J. Chambua, and Z. Y. Khan. 2019. *Multi-task learning model based on recurrent convolutional neural networks for citation sentiment and purpose classification.*

X. Zhu, P. Turney, D. Lemire, and A. Vellino. 2015. *Measuring academic influence: Not all citations are equal.*

Acknowledgements

This document has been adapted from the ACL Rolling Review Template (ACL ARR) by Min-Yen Kan. You may find the original template, which NUS has also contributed to in the past, here: `https://www.overleaf.com/latex/templates/acl-rolling-review-template/jxbhdzhmcpdm`. We have omitted much of the original document to cut down on verbiage.

Statement of Independent Work

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[2]. This particular document did not use any AI Tools to proofcheck and was constructed and edited purely by manual work.

If the production of your report used AI Tools (inclusive of Generative AI), do keep detailed logs of how you used AI Tools, as your project requires the accountability of an audit trail of your interaction(s) with such tools (prompts, output).

Signed,
A0201815R, e0415624@u.nus.edu,
A0219668M, e0550326@u.nus.edu,
A0220741U, e0555933@u.nus.edu,
A0222371R, e0559903@u.nus.edu,
A0248400N, e0925552@u.nus.edu,
A0275768H, e1127422@u.nus.edu

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

Appendix 1A

MLP architecture and hyperparameters

Feature	Input Size	Hidden Layer 1	Hidden Layer 2
POS Tags	12	15	12
Word Count	100	60	30
GloVe Embedding	200	120	60
All 3 features	212	120	60

Table 8: Size of MLP Hidden Layers

Optimizer	Adam
Learning rate	0.001
Epochs	30

Table 9: Training Hyperparameters

Appendix 1B

Feature Descriptions

Post Tag	The tokens in the input sentence were tagged with their Part-of-Speech (POS) using the nltk package and the count of each tag was used
Word Count	A sparse vector of the count of each token in the input was generated and Scalar Vector Decomposition was used to reduce it to a 100-dimensional vector
GloVe Embedding	The mean of the 200-dimensional GloVe embeddings of the tokens in the sentence was calculated to get a 200-dimensional vector. Out of vocabulary words were encoded as the mean GloVe vector across the entire vocabulary.

Table 10: Logistic Regression and MLP

GloVe Embedding	200-dimensional GloVe embeddings Out of vocabulary words were encoded as the mean GloVe vector across the entire vocabulary.

Table 11: RNN and LSTM

Appendix 1C

MLP performance against epochs

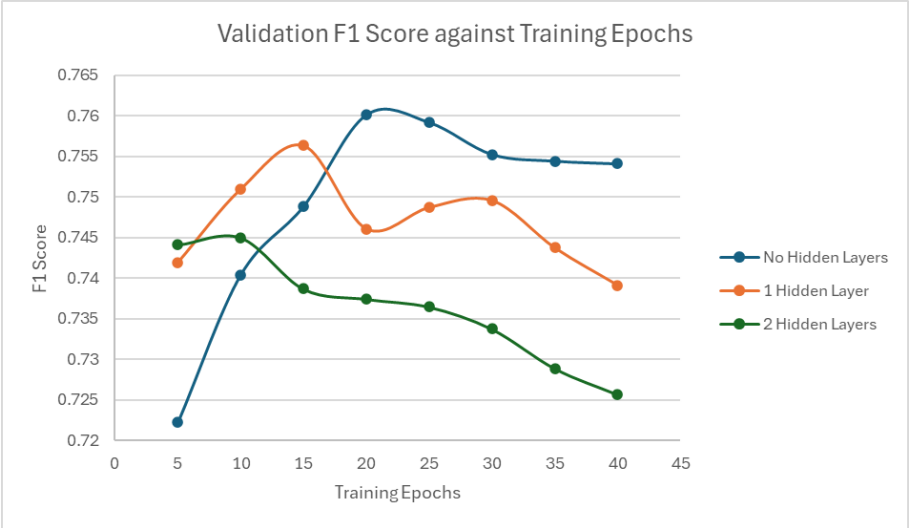

Figure 1: plot of MLP model performance across training epochs when all 3 features were used.

Appendix 1D

RNN Architecture

RNN	LSTM	BiLSTM
Embedding Layer [Vocabulary Size × 64]		
Dropout Layer [200]		
RNN Layer [200 × 64]	LSTM Layer [200 × 64	BiLSTM Layer [200 × 64]
Dropout Layer [200]		

Table 12: RNN Architecture

The dropout rate used was 0.3. ReLU was used as the activation function.

Appendix 1E

Feature attribution scores for various words

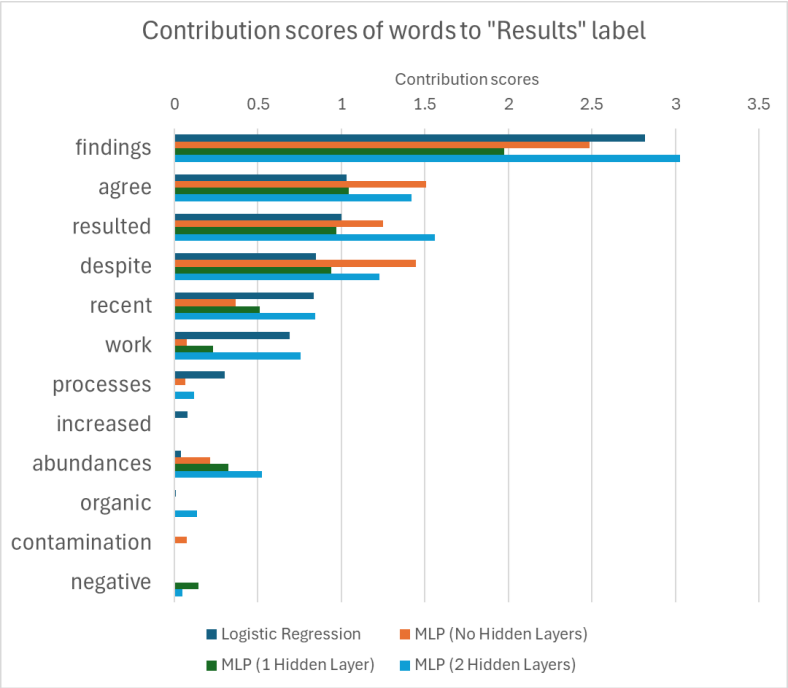

Figure 2: Plot of Contribution scores of words to "Results" label

Appendix 2

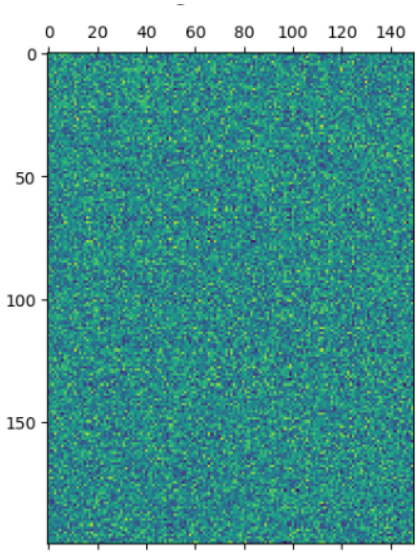

Figure 3: Mapping of word embeddings (vertical axis) to RNN hidden layer (horizontal axis)

Visualization of vanilla RNN matrix weights We charted the above to look for interesting relationships between the embedding dimensions and the hidden layer dimensions in a vanilla RNN model, in turn to develop more granular intuitions about how the model encodes semantic information. We note that there are no discernible patterns here.

Distribution of 'direct' and 'indirect' section names Based on the training dataset, we observe the following distribution of section names: Section names that do not contain any of the 3 intent labels: 70Section names that contain one of the 3 intent labels, but do not match the corresponding sample's label: 18Direct section names, i.e. those that contain the corresponding sample's label: 12

Experiments with activation functions Baseline model: RNN with attention, static features and full section names, without additional hidden layers.

- SwiGLU in combined layer
Epoch28: val loss=0.678, balanced accuracy=0.852, precision=0.875, recall=0.874, fscore=0.874
- Swish in combined layer
Epoch9: val loss=0.685, balanced accuracy=0.823, precision=0.869, recall=0.864, fscore=0.863
- SwiGLU after attention decoder
Epoch10: val loss=0.796, balanced accuracy=0.701, precision=0.803, recall=0.763, fscore=0.760

We observe that the RNN attention model performed marginally worse than the baseline with either activation function, implying that the gating effect of SwiGLU, as well as the non-monotonicity of both functions, do not value-add to what seems to be a rather thin body of existing features. With this and other trials, it appears that adding further complexity to feature representation leads consistently to overfitting, which suggests that the complexities among existing features have been quite exhaustively represented.

Interestingly, we also observe that all variants of the RNN model perform much more poorly if a softmax function is added as a final layer.

Appendix 3

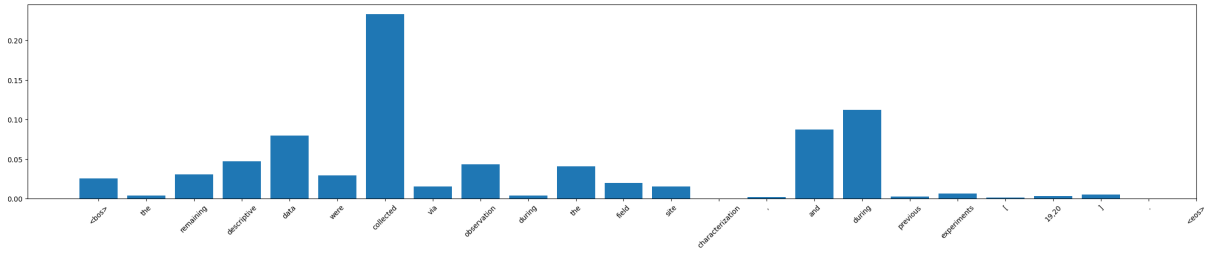

Figure 4: Attention weights for sample sentence 1

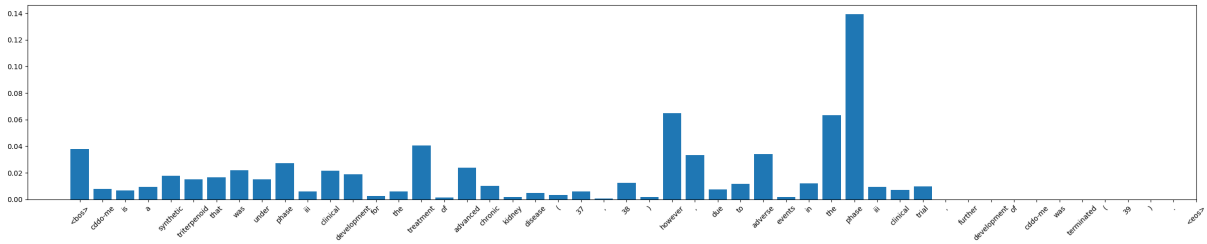

Figure 5: Attention weights for sample sentence 2

Validation and Test results for LSTM-based models				
Model	**Validation Data**		**Test Data**	
	Accuracy	**F1-Score**	**Accuracy**	**F1-Score**
LSTM (0.5)	0.7472	0.7110	0.7609	0.7250
LSTM with self-attention (0.5)	0.7982	0.7677	0.8017	0.7682
BiLSTM (0.5)	0.7834	0.7518	0.7985	0.7642
BiLSTM with self-attention (0.5)	0.8067	0.7825	0.8049	0.7751

Table 13: Validation and Test results for LSTM-based models

Modification / Metric	Loss	Balanced Accuracy	Precision	Recall	F1 Score
Control	0.669	0.849	0.882	0.882	**0.880**
Randomize all words (equivalently, n-gram, n=1)	0.700	0.814	0.860	0.854	**0.854**
Randomize n-grams, static n value (n=5)	0.692	0.821	0.865	0.861	**0.860**
Randomize n-grams, static n value (n=10)	0.676	0.856	0.881	0.881	**0.880**
Randomize n-grams, dynamic n value (n=m/5, m=length of sentence)	0.680	0.841	0.869	0.869	**0.868**

Table 14: Effects of modifications of the RNN with static features

Modification / Metric	Loss	Balanced Accuracy	Precision	Recall	F1 Score
Without static features (control)	0.459	0.756	0.804	0.806	**0.803**
With static features	0.697	0.820	0.849	0.850	**0.847**

Table 15: Effects of static features on RNN with attention

Modification / Metric	Loss	Balanced Accuracy	Precision	Recall	F1 Score
Without section name	0.697	0.820	0.849	0.850	**0.847**
With full section names	0.669	0.849	0.882	0.881	**0.880**
With only indirect section names	0.692	0.811	0.857	0.856	**0.853**

Table 16: Impact of section name on RNN with attention and static features

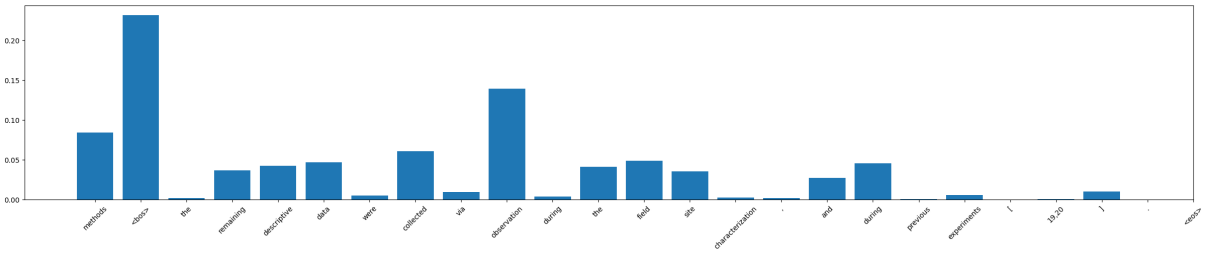

Figure 6: Attention vector with section name included

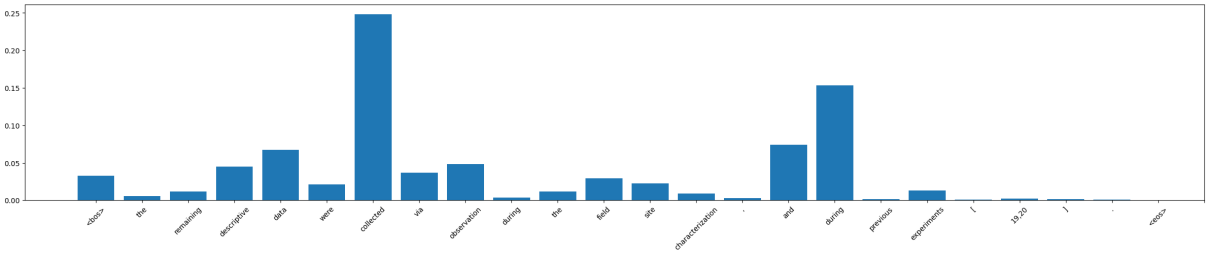

Figure 7: Attention vector with direct section name removed