CS4248 Final Report

A0236430N, A0236449W, A0245646X, A0275639M, A0275659J, A0275594M
Group 19
Mentored by Isaac Tay
{e0735359, e0735378, e0902052, e1127293, e1127313, e1127248, e1127248}@u.nus.edu

Abstract

The present study encompasses the experiments, findings, and evaluation of the application of various models towards the SciCite data. The models applied include Naive Bayes, Logistic Regression, CNN, LSTM, 4 variations of BERT, and GPT-2. From these, we derive multiple research discussions to obtain a deeper understanding of each model, especially in terms of its performance on semantic changes. Through our experiments, we uncovered distinct patterns in each model's efficacy in capturing semantic changes. Expanding upon our findings, we conducted an in-depth analysis of the underlying mechanisms that contribute to the varying performance of these models in handling semantic changes. By scrutinizing factors such as model architecture, training data diversity, and tuning strategies, we gained deeper insights into the nuances of semantic representation within each model.

1 Introduction

In the intricate landscape of scientific literature, citations not only measure scholarly impact but also chart the progression of research themes. The task of discerning the intent behind each citation, a process that remains largely manual, poses a significant bottleneck, limiting the efficiency of literature synthesis and systematic reviews. This project introduces an automated system, in the form of machine learning models, to classify the intent of citation sentences, pinpointing whether they provide background, reference methods, or compare results. On top of that, the project aims to have a deep understanding of how different models handle semantic changes and disturbances within text.

Our approach harnesses a carefully designed data preprocessing pipeline, essential for preparing the textual content for subsequent analysis. Through the exploration of advanced machine learning models, we set comprehensive benchmarks for citation intent classification. By leveraging traditional NLP techniques alongside state-of-the-art models like GPT, BERT, and its derivatives, our work provides a better understanding of the underlying mechanisms for all these models, specifically on how they handle semantic alterations and noises.

2 Related Work

The exploration of citation intent within the academic literature has increasingly become a topic of interest, building upon the early groundwork laid by seminal studies in the field of NLP text classification. The pioneering work of Zhang and Wallace (2015) established the initial frameworks for applying machine learning to citation analysis, which has since evolved through the integration of more sophisticated NLP techniques.

Recent advancements have been marked by the adoption of transformer-based models for text classification tasks, as evidenced by the contributions of Vaswani et al. (2017) in developing the attention mechanism that underpins such models. Our exploration of transformer-based architectures, including BERT (Devlin et al., 2018) and its derivatives, such as RoBERTa, ALBERT, and DistilBERT, is informed by the findings of these studies that demonstrate their superior performance on a range of NLP tasks.

A gap in the literature persists, however, in the nuanced application of these advanced models to the specific domain of citation intent classification. While researchers have applied BERT to the classification of citation intent with promising results, our project extends this line of inquiry by comparing a variety of BERT-related models, as well as incorporating other architectures like GPT (Radford et al., 2018) against the backdrop of the well-established SciCite dataset (Cohan et al., 2019), in a comprehensive and systematic study.

3 Corpus Analysis and Method

I. Exploratory Data Analysis

The initial stage of our analytical process focused on gaining a comprehensive understanding of the dataset's composition. Upon examination, the dataset was found to be complete, with no missing values present across all variables. This indicates a well-maintained and curated dataset, allowing us to proceed without the need for imputation strategies, thus preserving the integrity of our analysis.

An integral part of our initial data analysis was understanding the class distribution, as it directly informs the potential for class imbalance within the dataset. We visualized the distribution of the three citation intent classes: 'background', 'method', and 'result'. The resulting plot, as seen in Figure 1, revealed a significant imbalance with the 'background' class being the most prevalent.

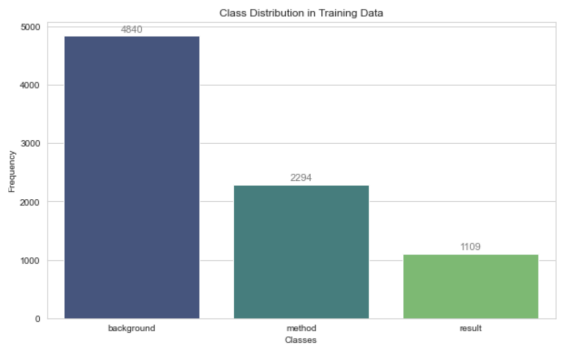

Figure 1: Class Distribution in Training Dataset

This imbalance highlights the importance of considering strategies such as upsampling the minority classes to ensure that our classifiers do not become biased towards the majority class. The skewed distribution also underscores the need for careful consideration of model evaluation metrics — accuracy alone may not suffice, and a more nuanced view like the F1-score becomes imperative.

II. Data Preprocessing

Data preprocessing is a pivotal component of our methodology, designed to convert raw text into a more analyzable form. Our preprocessing pipeline consisted of four distinct functions, each function could be applied independently or in sequence, aiming to refine the dataset and generate multiple versions for future experiments.

The first function focused on **removing stopwords**, which are common, low-information words in English that don't contribute significantly to classification tasks to emphasize more meaningful terms in the text. The second function involved **lemmatization**, merging various forms of a word into a single form for analysis, simplifying our data, and aiding our models in grasping the language's structure. **Tokenization** followed, by breaking text into tokens for detailed analysis. Lastly, **data augmentation** (**upsampling**) addressed class imbalances by artificially expanding the dataset, particularly boosting underrepresented classes, bolstering training data diversity, and enhancing model robustness against imbalances.

III. Feature Extraction

Feature extraction is a critical phase that transforms raw data into a structured format that models can exploit. In our project, we utilized three well-established techniques for data representation: Term Frequency-Inverse Document Frequency (TF-IDF), Bag of Words (BOW), and GloVe Embeddings.

TF-IDF is a powerful numerical statistic that gauges the significance of a word within a document collection, blending term frequency and inverse document frequency to assess both local and global term relevance, enabling prioritization of words pivotal for categorizing citations. **BOW** offers a straightforward count of word occurrences in a document, lacking in nuance but excelling in computational efficiency for text classification. Meanwhile, **GloVe** embeddings delve into semantic nuances by leveraging global word co-occurrence statistics across the corpus, though they demand more computational resources for generation.

IV. Models

To address the task of citation intent classification, we employed a two-tiered modeling approach: starting with basic models for establishing baselines, followed by an exploration of advanced deep learning architectures to push the boundaries of performance.

Multinomial Naive Bayes: We considered MNB with TF-IDF as a baseline model to predict citation intent based on probabilistic learning of word frequencies. Its ability to handle high-dimensional data makes it an excellent first step in our modeling process.

Logistic Regression: We employ Logistic Regression for its capacity to handle multiclass classification problems, leveraging its efficiency and simplicity. Logistic regression can effectively model linear relationships between features and target variables, making it a reasonable choice as a performance benchmark for classification tasks.

Convolutional Neural Network: CNN's ability to effectively capture local and positional patterns within text data through their convolutional layers may make them adept at identifying context and semantics in various text structures in this task.

LSTM: LSTMs are designed to handle sequential data, making them well-suited for tasks where the order of words or tokens in a sentence is crucial for understanding meaning and classification tasks. They can capture long-term dependencies in sequences by learning to selectively remember or forget information over time which is essential for understanding context and capturing semantic relationships in text data.

BERT-Base: A foundational model in the NLP community, BERT-Base was included for its bidirectional training and powerful contextual embeddings. BERT base has been widely used and extensively evaluated across various NLP tasks, pre-trained on a large corpus of text, demonstrating strong performance in text classification and language understanding tasks.

Roberta: A robustly optimized BERT variant pre-trained on an even larger corpus of text, Roberta refines the masked language model training of BERT for improved performance. It improves upon the BERT base by using a more extensive pre-training corpus and removing the next sentence prediction (NSP) task which leads to better generalization and understanding of language, outperforming BERT in text classification, question answering, and sentiment analysis tasks.

ALBERT: A lighter version of BERT that achieves competitive results with fewer parameters, reducing memory consumption, and improving speed. It introduces parameter-sharing techniques across layers such as cross-layer parameter sharing and factorized embedding parameterization, allowing for more effective use of parameters and reducing redundancy which improves the model's capacity to learn representations from limited data.

DistilBERT: A distilled version of BERT that retains most of the original model's performance but is smaller, faster, cheaper, and lighter. It provides a good tradeoff between faster inference time and reduced computational resource requirements while preserving over 95% of BERT's performance.

GPT-2: At the heart of GPT-2 lies its decoder-only transformer, which employs an auto-regressive approach, meaning it generates output tokens one at a time while conditioning previously generated tokens. GPT-2's ability to generate contextually relevant text can then be adapted for text classification.

4 Experiments

Our experimental framework was designed to evaluate the effectiveness of various algorithms in classifying citation intent.

I. Evaluation Metrics

Model performance was assessed using accuracy and F1-score metrics, where accuracy provides a direct measure of classifier success, and F1-score, balancing precision and recall, is particularly valuable in contexts with imbalanced data.

II. Model Training

Naive Bayes and Logistic Regression

Both models were trained and tested using two different data representations—TF-IDF and BOW. The TF-IDF vectorizer specifies a bi-gram range, sets minimum and maximum document frequency thresholds for vocabulary inclusion, and enables term frequency smoothing, resulting in a more noise-reduced feature set. Besides, both augmented and non-augmented data were evaluated to measure the impact on the model's performance. To optimize both models, we implemented GridSearchCV for hyperparameter tuning. We used brute force instead of tailored hyperparameter tuning because these 2 models function as a baseline or preliminary benchmark for all advanced models.

LLMs (RoBERTa, BERT-Base, DistilBERT, ALBERT, and GPT-2)

We first initialize these LLMs with the corresponding pre-trained model. Then, we systematically tailored the hyperparameter tuning for each LLM model. We first used the number of epochs = 1 and the learning rate of 1e-3. We noticed there is an optimal value for epoch and learning rate by tracking the training loss. Training with too few epochs may lead to underfitting, where the model fails to capture complex patterns in the data, and increasing the epochs can further push the training loss down. On the other hand, training for too many epochs increases the risk of overfitting, where the model memorizes the training data but performs poorly on new data, which is shown by low training loss but poorer validation performance.

Similarly, training with a small learning rate can lead to slower convergence and suboptimal performance despite large epochs. Conversely, large learning rates can lead to overshooting the optimal point and unstable loss descent. We proceed by tweaking the number of epochs and learning rate while plotting the validation loss and performance to find the sweet spot that maximizes model performance without overfitting or underfitting.

Because of the approach above, each LLM model that we implemented may use a different learning rate and epoch in the end. We set the batch size to 16, as a larger batch size may cause a memory limit error. Furthermore, we also created a custom dataset loader for both training and multiple test sets to evaluate the model's performance across diverse data and used the Adam or AdamW optimizer with a cross-entropy loss function to help the training process.

LSTM

For LSTM, we created a vocabulary that maps a word to an integer using the training data and build_vocab_from_iterator from torchtext library, and converted the text into numerical format using this vocabulary. The model architecture consists of a trainable embedding layer that is initialized using Glove embeddings, an LSTM layer, and the final dense layer, which converts the last LSTM output to logits. The model was trained using the Adam optimizer and cross-entropy loss for 10 epochs with a batch size of 32.

Convolutional Neural Network (CNN)

For CNN, we used GloVe embeddings for the embedding matrix in the model. The CNN architecture included convolutional layers with different filter sizes, batch normalization, max pooling for feature reduction, and dense layers with dropout for regularization. The model was compiled using the Adam optimizer and binary cross-entropy loss, with the metrics of accuracy and macro-averaging F1 score. The model was trained for 250 epochs with a batch size of 256, and performance was evaluated on various test sets

III. Experiment Main Results

Table 1 and Table 2 in the **Appendix** show the whole results of 9 models' performance on 7 categories of test data. The code can be viewed in our repository.[1] Full results table is available in this link.[2]

IV. Exploration of Semantic Changes

To evaluate the robustness and flexibility of the classifiers under varied textual conditions, we delineated six distinct categories. We filter the test data according to its physical or structural properties into 3 categories: **short data** for texts with no more than 25 words, **long data** for texts exceeding 25 words, and **paragraph data** for texts spanning multiple sentences. Then we transform the whole test data into 3 varieties: **typographical data** where we introduce deliberate randomized spelling errors within a word and randomly swap 2 adjacent words within a sentence, **synonym data** where we convert each word in a sentence to their direct synonyms (from nltk's wordnet synsets) without taking into account the context of the sentence, and **paraphrased data** where we use OpenAI API to paraphrase all sentences in the test data. We then have 6 additional datasets to test our models with.

VI. Explanation: Unpacking the Main Results

In this section, we will detail the reasons behind the performance outcomes we obtain from LSTM and LLMs (BERTs and GPT). We established MNB TF-IDF as our model baseline; higher accuracy and F1 than this baseline indicate better performance, and vice versa.

LSTM

Long data: Recall that LSTM is a specialized RNN that is implemented to tackle the vanishing gradient problem (Hochreiter & Schmidhuber, 1997), which is commonly present when a vanilla RNN is trying to process very long texts. Hence, it is expected that it should perform well on long data. However, this is not the case, as it performs even worse than the baseline MNB with TF-IDF. We believe it's not that LSTM can't model long-term dependencies; rather, it's because the implemented LSTM with GloVE may not be effective for the Scicite dataset.

Typo data: We aim to delve deeper into whether this is caused by typos within words or the swapping of words within sentences. We found the latter does not affect the accuracy and F1, whereas the former drops performance by 4%. LSTM can handle text with some word swapping in sentences due to their ability to grasp sequential dependencies and learn from word order, but excessive or random swapping can still challenge their predictions. Algorithmically, when inputs for adjacent LSTM cells are swapped, the output effects on subsequent cells are limited, whereas when inputs are swapped excessively between faraway cells, it may change the overall output since changes to faraway cells may be carried over by the LSTM memory cell state.

For typos within words, swapping 2 adjacent letters may alter the word embedding significantly because the embedding space is sensitive to small changes in the input, which then leads to embeddings that are semantically distant from their correct counterparts. As a result, the LSTM may struggle to accurately interpret the meaning of words with typos.

Paraphrased data: In the case of paraphrasing, we have to consider the chance where the input is invalid, for example, if the transformations are too complex or the produced output is of a dramatically different syntactic structure (Iyyer et al., 2018). On the other hand, it is also a possibility that LSTM is unable to detect the equivalence between two terms. With the simplicity of the LSTM model implemented, it may not have captured enough features, causing poor performance on paraphrased data.

LLM

Long data: LLM's good performance on long data is mainly due to its transformer capabilities

[1] https://github.com/nixonwidjaja/scicite

[2] https://docs.google.com/spreadsheets/d/1brg1w4YGOFVnPctY1Lgq-VDRvCcFfwbjvNshJaeUPaA/edit?usp=sharing

of attention and positional encodings. Attention allows the transformers to visualize the context of the inputs. Paired up with positional encodings, the transformers will then understand the position of the tokens in the sequence. The combination of these two features allows transformers to encapsulate sequence information. Kazemnejad (2023) states that larger context sizes allow a model to benefit from more in-context learning examples, higher numbers of reasoning steps, and longer text generation. We have 6940 long data and 1303 short data for training. These numbers definitely aid in the Transformer's ability to generalize and in turn, perform well with long data.

Typo data: Swapping adjacent letters in a word is one of the most common typos. Given the large-scale pre-training data in BERT (Devlin et al., 2018) and GPT-2 (Radford et al., 2019), we suspect that these models already saw a lot of these typos during pre-training. Hence, these models are robust against this type of typo. Recall also that the attention mechanism allows LLM to focus on the relevant part of the texts and effectively retrieve the information from the surrounding context. When two neighboring words are randomly swapped, it won't change the surrounding context significantly, and hence the LLM will still perform well in this case. This explains why LLMs perform well in typo data.

Paraphrased data: Transformers can predict paraphrased text well due to their attention and self-attention mechanisms, which allow them to capture semantic similarities and contextual information effectively. Furthermore, BERT's pre-trained embeddings are trained on a vast amount of text data, allowing it to capture rich semantic information about words and their contextual usage which means that words with similar meanings or in similar contexts tend to have embeddings that are close to each other in the embedding space.

5 Discussion

I. Why do CNN and LSTM perform worse than Logistic Regression and Naive Bayes?

1. Model Complexity vs. Data Complexity:
The advanced models such as CNN and LSTM are highly capable of modeling complex relationships due to their architecture. However, these models are also more susceptible to overfitting, especially when the model's complexity doesn't match the task's complexity. Moreover, the sequential nature of the data may not be as critical, making the complexity of LSTM unnecessary. Furthermore, deep learning models often require large volumes of diverse training data to generalize effectively. If the dataset at hand does not meet these requirements, it can limit the models' ability to perform optimally.

CNNs are adept at extracting spatial or sequential patterns from data but may not be optimized for citation data lacking such patterns, hindering their effectiveness. Moreover, CNNs require substantial data for feature extraction, posing challenges in citation classification with potentially limited data availability.

Meanwhile, logistic regression assumes a linear relationship between the features and the log-odds of the target variable. In citation classification, the text content might be effectively represented almost linearly, hence, logistic regression may model the relationships between these features and the citation classes more accurately than complex models.

2. Suitability of Text Representation:
The choice of text representation can significantly impact model performance. In our experiments, models using GloVe embeddings (CNN and LSTM) underperformed compared to those using TF-IDF (MNB and LR). This might indicate that, for our specific dataset and task, GloVe embeddings do not provide any additional value over TF-IDF. GloVe embeddings are pre-trained on extensive web-based text corpora, and may not align perfectly with the specialized academic language found in our dataset. This misalignment can lead to less effective text representation as the embeddings may not capture the specific semantic nuances

(Pennington et al., 2014). In contrast, TF-IDF emphasizes words that are important and unique to the specific corpus, potentially offering more relevant features.

3. Hyperparameter Tuning:
Optimal neural network performance often hinges on finding the right combination of hyperparameters. Improper tuning can lead to models that do not generalize well beyond their training data (Bengio, 2012). In our experiments, the suboptimal performance of CNN and LSTM models might stem from not fully optimizing these parameters. Tuning various parameters in LSTM and CNN is also a difficult task due to their complexity. Additionally, without adequate regularization strategies, these models can easily overfit the training data.

II. Why do LLM models tend to handle typos and paraphrased variations better than synonym data?

We first define synonym data, where each word of the input is converted to a synonym based on its POS tag. Hence, there may be some occurrences in which translated sentences do not make sense. With this in mind, the likelihood of breaking the contextual meaning of a sentence is pretty high. This is the challenge of handling synonymized data and thus, it is expected that most models deteriorate in performance when understanding said inputs.

O: "In addition, the <u>result</u> of the present study <u>supports</u> <u>previous</u> studies, which did not <u>**find**</u> increased rates of first-born <u>children</u> among <u>individual</u> with OCD (20,31,34)."
S: "In addition, the <u>effect</u> of the present study <u>back up</u> <u>old</u> studies, which did not <u>**happen**</u> increased rates of first-born <u>kid</u> among <u>single</u> with OCD (20,31,34)."

A comparison between **O**riginal and **S**ynonymized data

One of the most vital parts of the LLM architecture is the attention mechanism, which is what we believe to be the main cause of the deteriorating performance in processing synonymized data. The idea of the attention mechanism is to aid the model's learning of word positions and their context. Comparing the attention mechanism for typos and paraphrased data, the structures of sentences for these inputs are still well structured. In terms of context, the attention mechanism allows the model to understand typos and paraphrased sentences better. Conversely, synonymized data disregards the overall sentence structure and its meaning. Hence, the attention mechanism can be misled when estimating scores for the words of these types of inputs.

Not only for LLMs but in general, models will tend to worsen in handling synonym-substituted data, as the creation of this data does not ensure grammatical correctness (Chiang et al., 2023). At the same time, it is also important to consider that just one word is enough to change the intent of the sentence. Consequently, LLMs may struggle when facing synonymized substituted data.

III. Why do BERT models perform relatively worse than GPT-2 in our citation intent classification dataset?

We believe that this is likely caused by two primary reasons, the difference in model architectures and pre-training data as explained below.

1. Difference in Model Architectures:
BERT and its variants are built using transformer-based architectures designed specifically for bi-directional training of language representations. In BERT variants, only the encoder part of the transformer is used, which allows them to focus on learning the contextual understanding of a language that can then be used for text classification. On the other hand, GPT-2 is built using transformer-based architectures designed explicitly for auto-regressive text generation. Unlike BERT, GPT-2 only uses the decoder part of the transformer, which explains GPT-2 state-of-the-art performance in text generation. Despite that, the generative context created by GPT-2 can also be used to do text classification. For instance, we can simply add a linear classification step in the last layer of the default GPT-2 model. We suspect that in classifying the

SciCite dataset, the generative context of a text may play a more pivotal role than the language representation itself. This would explain why GPT-2 performs better than BERT and its variants in our citation intent classification dataset.

2. Difference in Pre-Training Data:
We only trained the GPT-2 and BERT models with 10 epochs at most, due to time and hardware limitations. Hence, the initial weights of each model are very important. If they are very far from optimal, it might be difficult to get a good model. In our implementation of GPT-2 and BERT, we initialize our classifier with pre-trained models. GPT-2 model is pre-trained using 40 GB of social media texts (Radford et al., 2019). On the other hand, Devlin et al. (2018) state that the BERT-base model is pre-trained using 3.3 billion words of data from the English Wikipedia and BookCorpus, which we approximated to be around 15 GB in size. Thus, GPT-2 is pre-trained with significantly more corpus data than that of BERT which explains why GPT-2 models outperform BERT.

IV. Why does Distilbert surpass BERT-base across all categories, despite being more lightweight?
Being more lightweight does not necessarily mean Distilbert is worse than BERT. In fact, Distilbert is proven empirically to retain 97% of BERT performance across different datasets (Sanh et al., 2020). The key to Distilbert's smaller, faster, and better performance compared to BERT is knowledge distillation, where a smaller, more lightweight model (called the **student model**) is trained to mimic the behavior of a more complex model (**teacher model**, in this case, BERT-base) by learning the teacher's generated probability distributions.

However, BERT should have been more robust than Distilbert, but why does Distilbert outperform BERT for this dataset? DistilBERT has the same general architecture as BERT, but the token-type embeddings are removed while the number of layers is reduced by half (Sanh et al., 2020). Token-type embeddings are additional embeddings used in BERT to differentiate between segments or sequences of text in a given input. Removing this embedding improves Distilbert's performance most likely because the features needed for citation classification do not depend on the segment location. Hence, this removal may allow Distilbert to focus on more relevant information and reduce the model's capacity to overfit irrelevant segment distinctions, leading to improved generalization.

With fewer layers, DistilBERT may generalize better to new data, leading to improved performance. This is because Distilbert as a less complex model might adapt more quickly to the target domain during fine-tuning. While maintaining sufficient representation capacity, Distilbert with its fewer layers outperforms BERT most probably because this dataset does not require deeper-level features captured by BERT to accurately predict the category classification.

6 Conclusion

This project investigated various machine learning models' ability to classify citation intent in the SciCite dataset, focusing on their response to semantic changes. Our findings revealed that emphasizing the need to match model complexity with task demands. Transformer-based models excelled at managing complex semantic alterations, performing well across different linguistic transformations.

However, the reliance on a single dataset may limit the generalizability of our findings, and the high computational demands of advanced models restricted their optimization. Future research should explore diverse datasets and develop methods to enhance the efficiency of transformer models, making advanced NLP technologies more viable for broader use. Overall, our study deepens the understanding of semantic variation handling in citation intent classification and suggests directions for enhancing model applicability and performance.

7 Appendix

Test Data	MNB TFIDF	MNB BOW	LR TFIDF	LR BOW	GloVE LSTM	GloVE CNN	Roberta	Albert	DistilBert	BERT-Base, Uncased	GPT-2
Default Test Data	0.75873	0.74906	0.77539	0.77700	0.73778	0.64804	0.83665	0.85008	0.85277	0.84148	0.85921
Long Data	0.75234	0.74358	0.77235	0.77235	0.73358	0.64290	0.80175	0.84615	0.85178	0.80550	0.85428
Short Data	0.79770	0.78244	0.79389	0.80534	0.76336	0.67939	0.80916	0.87405	0.85878	0.83969	0.88931
Paragraph Data	0.78450	0.74818	0.78692	0.80145	0.74818	0.64165	0.79903	0.84998	0.85956	0.81598	0.85714
Typo Data	0.75282	0.73455	0.75658	0.75228	0.71359	0.64105	0.79097	0.81354	0.82751	0.79044	0.83396
Synonym Data	0.68995	0.69854	0.66953	0.66469	0.66093	0.58517	0.76679	0.74960	0.78076	0.75121	0.80978
Paraphrased data	0.75389	0.74153	0.76034	0.76034	0.70500	0.63138	0.75819	0.82805	0.82160	0.76088	0.81730

Table 1: Accuracy score for experiments

Test Data	MNB TFIDF	MNB BOW	LR TFIDF	LR BOW	GloVE LSTM	GloVE CNN	Roberta	Albert	DistilBert	BERT-Base, Uncased	GPT-2
Default Test Data	0.72620	0.70441	0.74979	0.74725	0.70809	0.70731	0.80087	0.83034	0.83778	0.79672	0.84320
Long Data	0.71778	0.69694	0.74631	0.74165	0.70430	0.71177	0.75766	0.82584	0.83607	0.75217	0.83814
Short Data	0.77622	0.74811	0.77121	0.72426	0.73319	0.66362	0.76056	0.86011	0.85259	0.77279	0.87797
Paragraph Data	0.75257	0.70082	0.76611	0.77945	0.72449	0.75132	0.78419	0.82996	0.85115	0.77961	0.84117
Typo Data	0.71383	0.68237	0.72523	0.71715	0.67565	0.67345	0.73502	0.77306	0.80061	0.73635	0.81445
Synonym Data	0.59607	0.59748	0.55024	0.54610	0.56793	0.46597	0.69969	0.63794	0.70871	0.66497	0.76672
Paraphrased data	0.71912	0.69669	0.73212	0.72892	0.67462	0.69057	0.71023	0.80503	0.80437	0.70852	0.79924

Table 2: F1-score for experiments

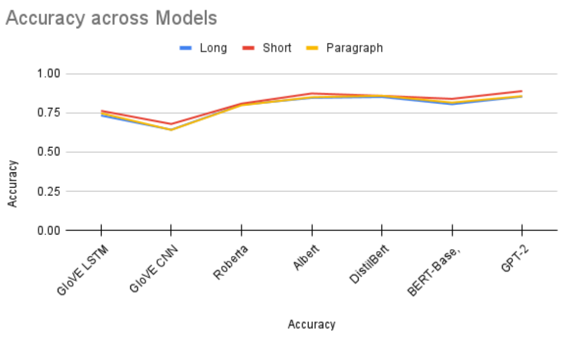

Figure 2: Accuracy score over advanced models on long, short, and paragraph test dataset

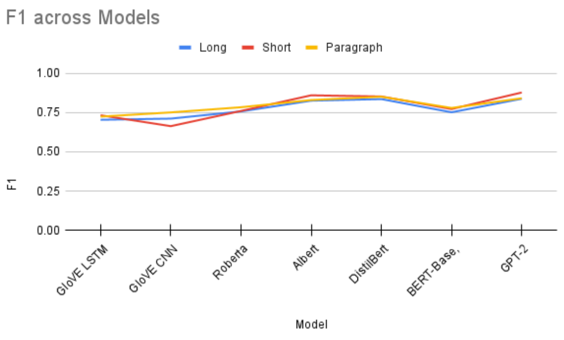

Figure 3: F1 score over advanced models on long, short, and paragraph test dataset

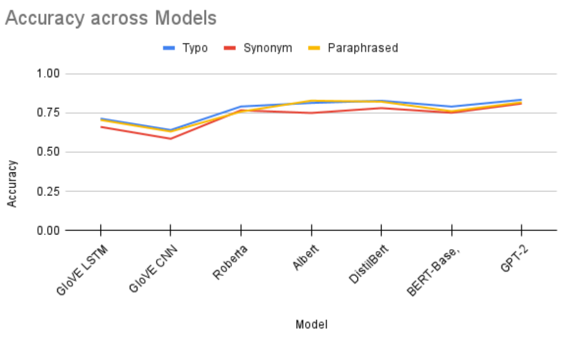

Figure 4: Accuracy score over advanced models on typos, synonyms, and paraphrased test dataset

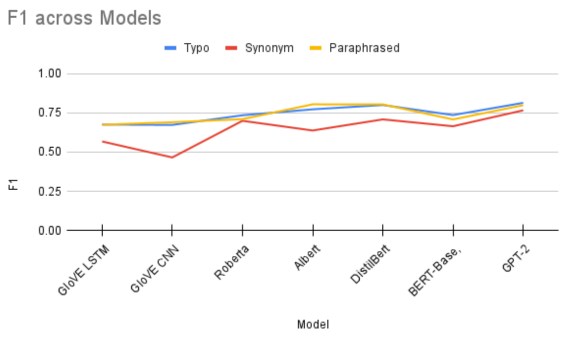

Figure 5: F1 score over advanced models on typos, synonyms, and paraphrased test dataset

8 References

Bengio, Y. (2012). Practical recommendations for gradient-based training of deep architectures. *ArXiv*. /abs/1206.5533

Chiang, C., & Lee, H. (2023). Are Synonym Substitution Attacks Really Synonym Substitution Attacks? https://aclanthology.org/2023.findings-acl.117.pdf

Cohan, A., Ammar, W., van Zuylen, M., & Cady, F. (2019). Structural Scaffolds for Citation Intent Classification in Scientific Publications *ArXiv*. /abs/1904.01608

Devlin, J., Chang, M., Lee, K., & Toutanova, K. (2018). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *ArXiv*. /abs/1810.04805

Hochreiter, S. & Schmidhuber, J. (1997). Long Short-term Memory. Neural computation. 9. 1735-80. 10.1162/neco.1997.9.8.1735.

Iyyer, M., Wieting, J., Gimpel, K., & Zettlemoyer, L. (2018). Adversarial Example Generation with Syntactically Controlled Paraphrase Networks. https://aclanthology.org/N18-1170.pdf

Kazemnejad, A., Padhi, I., Ramamurthy, K. N., Das, P., & Reddy, S. (2023). DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. *ArXiv*. /abs/2305.19466

Manning, C. D., Raghavan, P., & Schütze, H. (2008). Introduction to Information Retrieval. Cambridge University Press.

Pennington, J., Socher, R., and Manning, C. (2014). GloVe: Global Vectors for Word Representation. In Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 1532–1543, Doha, Qatar. Association for Computational Linguistics.

Radford A, Narasimhan K, Salimans T, et al. Improving language understanding by generative pre-training[J]. 2018.

Radford A, Wu J, Child R, et al (2019). Language Models are Unsupervised Multitask Learners.

Sanh, V., Debut, L., Chaumond, J., & Wolf, T. (2020). DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. *ArXiv.* /abs/1910.01108

Sun, S., Cheng, Y., Gan, Z., & Liu, J. (2019). Patient Knowledge Distillation for BERT Model Compression. *ArXiv.* /abs/1908.09355

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., & Polosukhin, I. (2017). Attention Is All You Need. *ArXiv*. /abs/1706.03762

Zhang, Y., & Wallace, B. (2015). A Sensitivity Analysis of (and Practitioners' Guide to) Convolutional Neural Networks for Sentence Classification. *ArXiv*. /abs/1510.03820

Zhou, Y., Alon, U., Chen, X., Wang, X., Agarwal, R., & Zhou, D. (2024). Transformers Can Achieve Length Generalization But Not Robustly. *ArXiv*. /abs/2402.0937

9 Acknowledgements

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. This particular document did not use any AI Tools to proof check 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,
A0236430N, A0236449W, A0245646X,
A0275639M, A0275659J, A0275594M
{e0735359, e0735378, e0902052, e1127293,
e1127313, e1127248,e1127248}@u.nus.edu