Zach Mickelson
Independent Research
5/5/2025

Does BERT Have the Goods?

An Exploration of BERT on Legal Document Classification

Background

The classification of legal texts plays a central role in legal research tools, particularly in systems like Westlaw's KeyCite system, where attorneys manually assign structured tags to opinions and other legal documents.[1] These systems help researchers navigate complex legal issues by linking documents to relevant legal principles. However, manual indexing is time-intensive, costly, and difficult to scale across large amounts of legal material. Over the past few years, there have been several advancements in transformer-based models have shown promise for natural language processing ("NLP") tasks. These models are capable of capturing the semantic nuance and specialized vocabulary of legal texts, offering a potential path toward automating legal issue classification and assisting editorial systems like KeyCite.

This project explores the viability of using Legal BERT to classify individual paragraphs from Georgia Appellate decisions dealing with the Uniform Commercial Code Article 2, which governs the sale of goods. By working at the paragraph level rather than the full-document level, this study tests the model's ability to perform fine-grained legal issue classification.

To assess the model's performance under different levels of label specificity, the dataset was annotated using three sets of categories: one with 53 specific labels (see

[1] Thomson Reuters, *KeyCite*, https://legal.thomsonreuters.com/en/products/westlaw/keycite (last visited May 4, 2025).

Appendix 1A), one with 19 broader categories (see Appendix 1B), and one with two general classes (UCC or not UCC). The aim was to evaluate how model accuracy responds to varying degrees of granularity in a classification setting. A total of approximately 5,100 paragraphs were manually labeled for this purpose.

Data

The case law used for this project was collected from Harvard's Case Law Project (hereinafter, the "Project").[2] All Georgia Appellate Court cases were downloaded from the Project and stored locally, recreating the directory structure of the original dataset.[3] These cases are stored in PDF and HTML formats and are divided into 342 volumes (essentially, folders). For the purposes of this research, the HTML format was selected for parsing and analysis.

Once the Project's directory was stored locally, Beautiful Soup was used to parse the HTML text.[4] The Project stored the cases with several classes,[5] which were extracted to construct the fields of the data frame. The extracted classes were "parties" (the case

[2] The Case Law Project is a store of the entire Harvard Law School Library's physical collection in a machine-readable format. The Case Law Project can be found here. https://case.law/
[3] This is the is the store of Georgia Appels Reports stored by the Project. It has all Georgia Appeals Reports between 1906 and 2017. https://case.law/caselaw/?reporter=ga-app
[4] Beautiful Soup is a Python Package that specializes in working with html data. A documentation for this package can be found here.
[5] Class attributes are basically HTML labels within the code sections which serve as markers. Classes allow a user to access specific elements within the data. Mozilla Contributors, *class* - HTML, MOZILLA DEV. NETWORK, https://developer.mozilla.org/en-US/docs/Web/HTML/Reference/Global_attributes/class (last visited May 5, 2025).

name), "decisiondate" (the date the opinion was issued), [6] and "opinion" [7] (the full text of the opinions)."[8] The "opinion" class was then split by the new paragraph characters. This information was then placed within a Pandas data frame with the following columns: "folder," "file," "case_name," "date," and "maj_op."[9] Each row represents a single paragraph within the case. Finally, a column was derived from "maj_op," which is between 2 and 3 paragraphs long. This was formed by taking the paragraphs around the "maj_op" paragraph in the entire case. For more information concerning these fields, see the Data Dictionary in Appendix 2.

For the scope of this project, only cases where sections of Uniformed Commercial Code Article 2 (UCC Article 2) were considered. The decision to focus on UCC Article 2 was based on several factors: (1) the UCC has been widely adopted in the United States;[10] (2) Article 2 deals with contracts for the sales of goods so the law is concrete in its application; and (3) although unknown initially, I suspected that contract disputes would be prevalent enough to form a sufficient sample size for training purposes.[11]

[6] The dates of the cases were collected in the format [Month Proper Name] [DD], [YYYY].
[7] The class "opinion" did not differentiate between minority and majority opinions. Only majority opinions were considered in this research. The majority opinions were selected by only collecting the first opinion in every case into the data frame.
[8] The class "citation" was considered; however, this class was irregularly stored; older cases prior to the 1980's were missing this field altogether.
[9] This process can be seen in the framebuild.ipynb in the deliverables folder.
[10] All states have adopted some parts of the UCC. 14 have adopted Article 2. https://uniformcommercialcode.uslegal.com/states-adopting-the-ucc/
[11] The suspicion was also grounded in the knowledge that Article 2 was adopted by the State of Georgia in 1951.

The data frame was filtered to include only those cases where UCC Article 2 was at issue. This was accomplished by matching keywords are the UCC Article II statutes in both the O.C.G.A. and the Georgia Code of 1933. The complete list of these keywords can be found in Appendix 3 and in the sterms.txt file, which is included in the deliverables. The in operator[12] was used for this purpose, so if a reference to any of the statutes in sterms.txt appeared in a case's majority opinion, the case was considered. The filtered data frame used in the modeling may be found in the file "filter_par2.csv."

Once the frames were completed, the "maj_op" columns were read manually. They were categorized using a list of UCC Article II related statutes and doctrines. These categories may be found in the cats_0.txt and the cats_1.txt in the deliverable file. The "cats_0.txt" file is a collection of 53 total categories, which are represented by a number and an associated doctrine. The "cats_1.txt" file has 19 total categories and is essentially written as a dictionary, using the cats_0.txt category as the key and the new category as the value. These new values are essentially collapsed of the cats_1 categories. For instance, cats_0.txt has seven categories dedicated to separate implied warranties (i.e the implied warranty of merchantability, the implied warranty of fitness for a particular purpose, etc.), while cats_1.txt has all implied warranty related topics under one category.

[12] The in operator is used to check if a value is a part of a collection of items. More information about Python's in operator can be found here. Dan Bader, *Python "in" Operator: How to Use the 'in' Keyword in Python*, Real Python (Dec. 14, 2022), https://realpython.com/python-in-operator/ (last visited May 5, 2025)

For more information, Appendices 1A for cats_1.txt and 1B for cats_1.txt, or the files directly in the deliverable folder.

There are three category columns in the filtered data frame: "cats_0," which was written manually using cats_0.txt, "cats_0," which was made iteratively using cats_1.txt,[13] and finally "UCC" was made iteratively by checking whether "cat_1" was a non-zero value or a zero value.[14] These are the categories used for labeling in the modeling processes. Since the paragraphs were labeled manually for the finest granularity, all rows in the filter_par2.csv file are not labeled. As of this writing, the data frame has 12,978 rows, and of these, 5073 are labeled.

Methods

Several different methods were tested on the data with all methods relying on neural network classification models. Most of the experiments utilized BERT architecture. BERT, or Bidirectional Encoder Representations from Transformers model, is a type of neural network architecture that is often employed in text classification tasks.[15] What distinguishes BERT from other NLP models, such as word2vec, is its

[13] This was done by essentially changing the cats_1.txt into a dictionary which used the original cats_0 classes as a key, and the new cats_1 classes as values. A for loop was then used to write the cat_1 column. This can be seen in the framebuild.ipynb file at lines
[14] This column was derived from the "cats_0" column was well. It is Boolean, and a 1 just denotes a non-zero value in the "cats_0" column.
[15] Cameron Hashemi-Pour & Ben Lutkevich, *BERT Language Model*, TechTarget SearchEnterpriseAI (Feb. 2024), https://www.techtarget.com/searchenterpriseai/definition/BERT-language-model (last visited May 5, 2025).

bidirectional processing of text. Traditional NLP models generally process text sequentially, analyzing words one at a time, whereas BERT can analyze a word in the context of the words before and after it, making it effective at evaluating context.

BERT is pretrained on two tasks: mask language modeling ("MLM"), and next sentence prediction ("NSP"). [16] MLM predicts hidden tokens[17] in a sequence, but the model has full access to the tokens to the left and right of the masked token, so it may analyze them in both directions.[18] For the purposes of this research, MLM was used, where some words in a sentence are masked, and the model attempts to predict them using the surrounding context.

For these experiments, Legal BERT was used. Legal Bert is a variant of BERT fine-tuned on 12 gigabytes of English legal texts (such as legislation, court documents, and contracts).[19] The version used in these experiments is the Legal BERT base uncased.

The multiclassification experiments were also performed on two versions of the text to see if there was a difference in performance. As mentioned previously, the

[16] Id.
[17] Tokens in this context is a way to break down text, so that a machine can analyze them. For instance, most tokenization techniques would, for instance, take a sentence and form a comma delineated list of words and store them. For more, here's a good explanation. DataCamp, *What Is Tokenization? Types, Use Cases, Implementation*, https://www.datacamp.com/blog/what-is-tokenization (last visited May 5, 2025)
[18] This is a good example of the usage for the MLM capabilities. Hugging Face, *Masked Language Modeling*, Hugging Face Transformers Documentation, https://huggingface.co/docs/transformers/en/tasks/masked_language_modeling, (last visited May 5, 2025).
[19] Ilias Chalkidis et al., *LEGAL-BERT-BASE-Uncased*, Hugging Face, https://huggingface.co/nlpaueb/legal-bert-base-uncased (last visited May 5, 2025).

dataframe has two text columns: "maj_op," which is a single paragraph within the appellate case, and "wcontext," which is the "maj_op" text and the neighboring paragraphs.

Beyond BERT, a set of binary classification models were also tested to determine if a model could distinguish between UCC-related and non-UCC related paragraphs. For these experiments three architectures were used: (1) A simple sequential feedforward model with an embedding layer, (2) a Long Short-Term Memory (LSTM) network, and (3) Legal BERT. The objective of these binary classification experiments was to assess the ability of a model to filter out unresponsive paragraphs before categorizing those that are relevant to the UCC as a potential method of addressing class imbalance issues.

Neural Network Architectures:

1. Sequential Feedforward Neural Network:

   A basic architecture used in the Boolean experiments consisted of a feedforward neural network. This architecture involves an input layer, several hidden layers, and an output layer.[20] Each hidden layer contains nodes that perform activation functions on the data, and every node in one layer is connected to every node in the subsequent layer. These types of networks can approximate complex relationships between input data and output predictions. The neural network was

[20] IBM, *What is a Neural Network?*, IBM (Oct. 6, 2021), https://www.ibm.com/think/topics/neural-networks#:~:text=Every%20neural%20network%20consists%20of,own%20associated%20weight%20and%20threshold (last visted May 5, 2025).

trained using an Adam optimizer, with a ReLU function in the hidden layers, and a sigmoid activation function for the output layer. [21]

2. Long Short-Term Memory (LSTM) Network:

An LSTM network is a type of recurrent neural network ("RNN") designed to handle sequential data. Unlike traditional neural networks, LSTMs have feedback loops (recurrent connections), enabling them to capture temporal dependencies. LSTMS are also equipped with Forget Gates, which allows the model to discard irrelevant information during training. [22] For this project, a Bidirectional LSTM was used, meaning the input sequence was processed in both forward and backward directions to capture context from both sides of each token.

Experiments

1. Multiclassification Legal BERT Models

A. Legal BERT base uncased (53 Classes, Class 0 Considered).

The first experiment used the cat_0 column as labels, with 53 categories in total. The text from the wcontext column was preprocessed by removing

[21] The formulas for relu and sigmoid are: ReLU-- f(x)= max(0,x); sigmoid: f(x) = 1/(1+e^x^-1). Pragati Baheti, *Activation Functions in Neural Networks [12 Types & Use Cases]*, V7 Labs (May 27, 2021), https://www.v7labs.com/blog/neural-networks-activation-functions (last visited May 5, 2025).

[22] Here's a good primer on the LSTM architecture. Shipra Saxena, *What is LSTM? Introduction to Long Short-Term Memory*, Analytics Vidhya, https://www.analyticsvidhya.com/blog/2021/03/introduction-to-long-short-term-memory-lstm/ (last visited May 5, 2025).

stop words and converting the text to lowercase. The dataset was split into training validation, and test sets with a 64/16/20 split (80/20 split twice, with a random seed of 0). The maximum token length was set to 512, which is the maximum accepted by BERT. The model was compiled using an Adam optimizer and sparse categorical cross-entropy as the loss function. After 50 epochs, no significant improvements in training or validation accuracy were observed, and training was halted. The model is saved as trained_model_f50.keras, and the code for this model can be found in model_1_cat_1.ipynb in the Deliverables. See Fig. 1 in the Appendix.

B. Legal BERT Uncased (53 Classes, Category 0 Dropped, 50 Epochs)

The model was similar to the previous one but excluded category 0 (the majority class) to address data imbalance. Despite experimenting with various oversampling and undersampling techniques, no substantial improvement was achieved. Two versions were run: one using the maj_op column and the other using the wcontext column. These models are saved as trained_model2_f50.keras and trained_model3_f50.keras, respectively. The code for these models can be found in model_2_cat_1.ipynb in the Deliverables folder.

C. Legal BERT Base Uncased (19 Classes, 50 and 25 Epochs)

In these experiments, the cat_1 column with 19 categories was used. Two versions were run: one including category 0 (trained for 50 epochs) and one excluding category 0 (trained for 25 epochs). The second model was run only for 25 epochs because evidence of overfitting was seen after 25 epochs (gain in loss function and failure to improve accuracy in the validation set). See also Fig. 6. The code for these models can be found in Model_4_cat_2.ipynb in the Deliverables folder.

2. Binary Classification Models

A. Feedforward binary classification model (10 epochs)

For this binary classification experiment, the maj_op column was preprocessed by removing stop words and converting text to lowercase. The text was tokenized and padded into sequences of 7,154 tokens (the maximum paragraph length). The UCC column was used as the binary label (0 or 1). The dataset was split into a 80/20 train/test ratio. The model used a feedforward neural network architecture, with a ReLU activation for the hidden layer and a sigmoid activation for the output. The binary cross-entropy loss function was used for evaluation. The code for all of the binary classification models can be found in model_3_binary.ipynb in the Deliverables folder.

B. Bidirectional LSTM Binary Classification Model (10 epochs):

The preprocessing steps were similar to the feedforward model. The UCC column was used as the binary label, and the data was split using an 80/20 ratio. The model consisted of an embedding layer, a bidirectional LSTM layer (with a dropout value of 0.5), and a dense layer with a sigmoid activation function. The binary cross-entropy loss function was used, and the model was trained for 10 epochs.

C. Legal BERT Uncased (2 Classes, 3 Epochs):

This model used the UCC column as the label (with two classes). The preprocessing steps were identical to the other BERT models. The loss function was binary cross-entropy, and the model was trained for 3 epochs.

Results

This section presents the performance outcomes of both the multiclass and binary classification experiments. Models were evaluated using accuracy, macro-averaged precision, recall and F1 score. Due to significant imbalance in the multiclass data, macro-averaged metrics are emphasized to capture performance across all classes, but tables outlining each individual model are available in Appendix 5 for review.

1. Multiclass Classification Results

Three sets of multiclass classification experiments were conducted using the Legal BERT base uncased model. The models varied in the number of categories (53 or 19) and the textual source used (wcontext or maj_op). The 1C models used the maj_op

text column, selected based on improved accuracy observed in the 1B models when compared to their wcontext counterparts. Key performance metrics for selected models are summarized in Table 1.

Table 1: Macro Metrics – Multiclass BERT Models

Model ID	Input	Zeroes Dropped	Label Set	Classes	Accuracy	Macro Precision	Macro Recall	Macro F1 Score
1A	wcontext	No	cat_0	53	0.726	0.243	0.148	0.172
1B	wcontext	Yes	cat_0	52	0.337	0.301	0.205	0.220
1B	maj_op	Yes	cat_0	52	0.372	0.197	0.182	0.171
1C	maj_op	No	cat_1	19	0.761	0.342	0.203	0.232
1C	maj_op	Yes	cat_1	18	0.361	0.275	0.204	0.200

Model 1A was trained using the wcontext input and the full 53 class label set with category 0 included. It achieved an accuracy of 72.6% and a macro F1 score of 0.172.

The two 1B models were trained using the same 53-category label set, with the only difference being the textual input source: wcontext versus maj_op. The model using the maj_op column slightly outperformed the wcontext model in terms of accuracy (37.2% vs. 33.7%) and macro F1 score (0.171 v. 0.220).

More notably, the maj_op 1B model demonstrated better performance across the minority classes, producing non-zero precision, recall, and F1 scores in a larger number of categories. For example, it achieved non-zero

F1 scores for categories such as Category 12 (F1 = 0.45), Category 14 (F1 = 0.67), and Category 34 (F1 = 0.47). In contrast, the wcontext model yielded zero precision and recall for most classes with particularly sparse performance outside of the dominant category 0.

Model 1C using the maj_op text and the 19 category cat_1 label set (with zeroes retained) achieved the highest overall accuracy (76%) and the best macro F1 score (0.232) among all of the multiclass models. However, like the 1A model, its performance was disproportionately driven by its success in classifying Category 0 (F1 = 0.86), which made up more than 70% of the test set. Performance across the other categories was varied, with several categories receiving zero precision and recall.

By contrast, the version of 1C trained without Category 0 (18 total classes) saw an overall accuracy drop to 36%, and its macro F1 score fell slightly to 0.200. Howeve3r, it showed a much stronger performance in some underrepresented classes, such as Category 2 (F1= 0.52) and Category 11 (F1 = 0.47).

However, across all models, performance varied significantly by class, with many minority classes receiving zero precision and recall. This is reflective of both substantial class imbalance, which disproportionately affected macro-averaged metrics, and the limited number of examples available for

certain categories. A full classification report for each model is included in Appendix 5.

Additionally, when specifically looking at how the model handled the additional context paragraphs in the "wcontext" column, the models appear to have a better accuracy without the context.

2. Binary Classification Results

Binary classification models were trained to distinguish between UCC-related paragraphs and all other text. Three architectures were tested: a feedforward sequential neural network (FNN), a bidirectional LSTM, and a Legal BERT classifier. Key metrics are shown in Table 2.

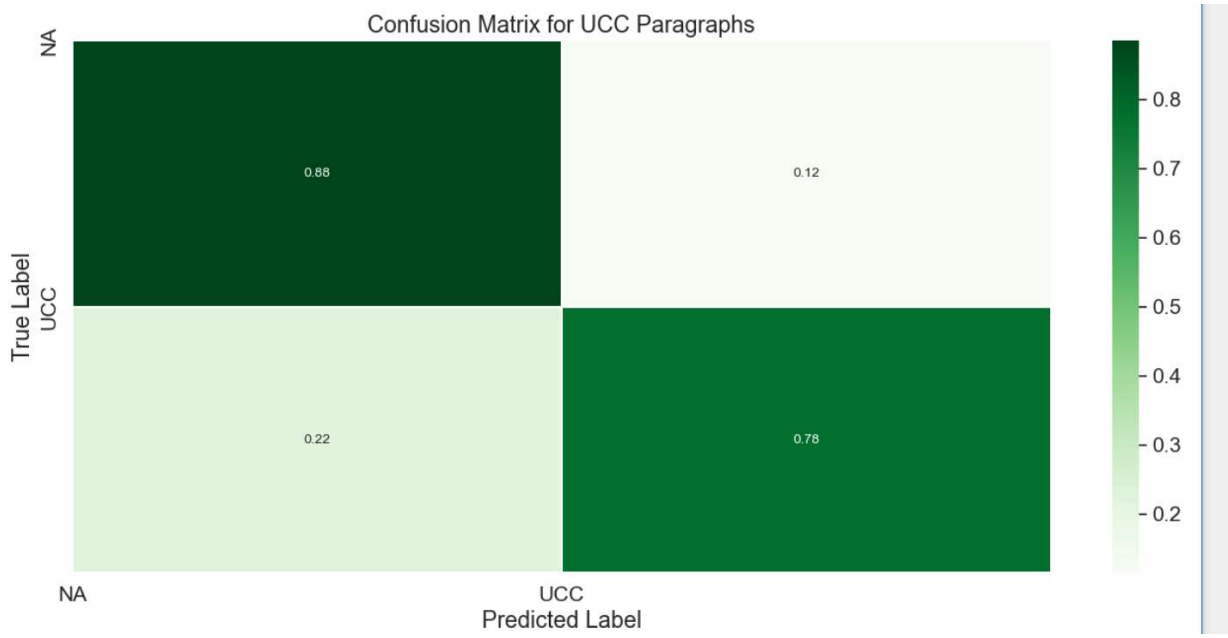
Table 2. Binary Classification Model Performance

Model	Accuracy	Precision	Recall	F1 Score
Feedforward	0.855	0.821	0.833	0.827
LSTM	0.833	0.709	0.691	0.700
Legal BERT	0.715	0.857	0.502	0.627

The FNN achieved the highest overall accuracy (85%) and F1 Score (0.827). Despite its architectural simplicity, it was most effective in distinguishing UCC-related content. Legal BERT showed high precision (0.857), indicating strong performance on true positives, but its lower recall (0.502) suggests it frequently failed to identify UCC-related paragraphs. For performance reports for the LSTM and BERT, see Appendix 5B.

Figure 1 displays the confusion matrix for the feedforward sequential model, which demonstrates balanced classification performance across the two classes. The FNN was able to correctly identify UCC related paragraphs 78% of the time.

Figure 1.



The performance results across the binary classification models underscore the varied effectiveness of different architectures in handling class imbalances and distinguishing UCC-related content. While the simpler FNN model performed well overall, the BERT-based model showed high precision but struggled with recall. These findings raise important considerations about the choice of model and the impact of architecture on handling data imbalance. In the Discussion section, we will explore the

implications of these results, focusing on model selection, the limitations of architectures used, and strategies to address challenges such as class imbalance.

Discussion

The classification results across both multiclass and binary tasks reveal important insights about model complexity, data characteristics, and the effects of different input sources. In particular, the multiclass models, especially those trained with the 53-class label set (Models 1A and 1B), struggled with effectively representing the minority classes. This led to a lower macro average F1 score, especially when the wcontext text input was used. Model 1A, which retained zero values, achieved an accuracy of 0.726 and macro F1 of 0.172, indicating that despite relatively good overall accuracy, performance across minority classes was limited (with macro-precision at 0.243 and macro-recall at 0.148). This model's inability to generalize across classes beyond the dominant categories likely stemmed from its overreliance on Category 0 ("other" category).

The two 1B models, trained with the same 53 category label set and both removed the zero, differed primarily in the textual input source: wcontext versus maj_op. Notably, the model trained with maj_op outperformed the wcontext version in terms of overall accuracy, achieving 37.2% compared to 33.7%. However, a closer examination reveals that although maj_op improved accuracy, maj_op version exhibited a significant drop in performance across several metrics, particularly in its ability to identify some minority classes. This is highlighted by the maj_op versions' lower macro F1 score (0.171 vs.

0.22), suggesting it was less effective at capturing the diversity of the categories, particular ones with low sample sizes.

Interestingly, while maj_op outperformed wcontext in several specific classes, such as Category 12 (F1: 0.45 vs. 0.30) and Category 15 (F1: 0.69 vs. 0.36), its lower overall lower macro suggests less consistent performance across all categories. The w context version showed stronger results in some other classes, such as Category 34 (F1: 0.55 vs. 0.47), and generally achieved a better balance between precision and recall. These results suggest that the richer context provided by the wcontext text input may have help the model recognize a broader variety of class signals, even if it had a detrimental effect on accuracy.

In summary, the maj_op version of Model 1B produced higher accuracy, but its reduced performance indicates weaker generalization across the full category set. The wcontext version, on the other hand, captured the class diversity more effectively, particularly for complex or underrepresented categories.

While the maj_op version performed better in terms of raw accuracy, the wcontext model was more balanced in its performance across underrepresented categories.

Model 1C models, which used the 19 category cat_1 label set and maj_op as a text input, differed only on whether they considered category 0. When Category 0 was retained, the model achieved high accuracy (0.7611), but this came at the cost of a low macro F1 score (0.171). This disparity reflects the model's heavy reliance on predicting

Category 0, which dominated the label distribution and itself achieved an F1 score of 0.86—skewing performance toward the majority class and masking poor results on minority categories.

Excluding category 0 caused overall accuracy to drop sharply to 36.14%, but macro F1 improved slightly to 0.200, and performance on several minority classes became more meaningful. For instance, category 4's F1 score increased from 0.18 to 0.52, category 11 improved from 0.32 to 0.47, and Category 2 rose from 0.40 to 0.52. These gains suggest that removing the overwhelming influence of Category 0 allowed the model to better distinguish and learn from less-represented categories.

In the binary classification task, simpler architectures outperformed more complex ones. The FNN achieved the strongest overall performance, with an accuracy of 0.855, F1 score of 0.827, and well-balanced precision (0.81) and recall(0.835). The LSTM model, while performing reasonably well with an accuracy of 0.832 and F1 of 0.700, lagged behind the FNN on both precision and recall. In contrast, BERT, despite achieving an F! score comparable to the FNN (0.826) suffered from a significantly lower recall (0.502), indicating that it missed many Article 2 related paragraphs even as it maintained high precision (0.857). These results suggest that, for this task, model simplicity may matter more than architectural sophistication, particularly in settings where the class boundaries are not deeply contextual.

A key challenge in this project stems from the nature of the data and labeling strategy. Many paragraphs contained multiple overlapping legal issues but were labeled

with a single label using the most dominant theme. This single-label constraint likely introduced ambiguity, limiting BERT's ability to generalize and accurately capture the full legal content. In practice, a model might correctly identify a secondary or tertiary issue within a paragraph but still be penalized for not predicting the dominant label. These conditions highlight a mismatch between the labeling scheme and the true complexity of the data at issue that was not fully anticipated when labeling began. Additionally, many cases in the corpus involved both UCC Article 2 and Article 9, which governs secured transactions and secured interests in personal property. The overlapping nature of these legal issues could have contributed to further labeling confusion, potentially affecting the model's ability to distinguish between them effectively.

Finally, less than half of the collected dataset is labeled as of this writing, which undoubtedly contributed to low sample sizes for many categories. These limitations certainly had an effect on performance, particularly in the multiclass tasks, and point to the need for further dataset development before more definitive conclusions can be drawn.

Next Steps

Upon reflection, several improvements could significantly enhance the classification performance and model reliability in future iterations of this project.

First, completing the labeling of the remaining dataset is a foundational step. A fully labeled corpus would certainly help address class imbalance, improve representation

for minority categories, and enable more reliable evaluation of model performance. This will also provide a more comprehensive understanding of the model's effectiveness across the entire dataset.

Second, there are two promising options that could meaningfully refine the annotation strategy and model performance. One potential improvement is to shift the labeling unit from paragraphs to individual sentences. This finer granularity would help reduce the overlap of multiple legal issues within a single example, making the classification task more focused. By isolating specific legal issues, we can offer the model cleaner input data, which could reduce ambiguity, particularly in longer, more complex paragraphs. This change would also make it easier to evaluate the model's performance on specific legal issues in isolation.

Alternatively, adopting a multi-label approach would certainly better reflect the real-world structure of legal texts. Rather than requiring a single dominant issue, multi-label models can accommodate the presence of multiple, co-occurring legal categories in the same paragraph. This would allow both the annotation process and the modeling pipeline to more faithfully capture the complexity of the reasoning. Collectively, these next steps aim to boost model performance and better align the modeling strategy to the inherent complexity of the legal domain.

Conclusion

This study explored the use of neural network classification models, particularly BERT architectures, for classifying Georgia Appeals Court decisions. The experiments demonstrated that while Legal BERT showed promise in the multiclass tasks, the performance was influenced by several factors, including class imbalance, data granularity, and the labeling strategy. The reliance on the majority class, especially in the multiclass models, often led to suboptimal performance for minority categories. Additionally, the use of the "maj_op" and "wcontext" columns as input sources provided insightful comparisons, with the latter offering richer context but leading to less consistent model performance.

The binary classification models highlighted the effectiveness of simpler architectures, with the feedforward neural network achieving the best overall performance. This suggests that, in certain tasks, simpler models can outperform more complex ones. However, sophisticated models like Legal BERT still showed value in terms of precision scores, albeit with limitations in recall. Moving forward, key future steps include the completion of dataset labeling, and exploring sentence-level labeling and/or adopting a multi-label approach. I expect these changes will be critical in improving model performance and achieving more reliable and accurate results.

Appendix

1. Data Dictionary (Categories / Labels):

A. cats_0.text

The following are the categories used for the labels in the data frame. They each represent either a single legal doctrine or statutory rule, or a combination of related legal doctrines or statutory rules. There are 53 total categories, represented by a single number from zero to 52. These categories can be found in the **cats_0.txt file** within the submission, but a brief description of the label can be found here.

0: other – This is a general category for paragraphs that are unrelated to any UCC Article 2 doctrine.
1: promissory estoppel / detrimental reliance
2: trade usage / practice
3: identification
4: implied warranty of merchantability
5: implied duty of good faith and fair dealing / bad faith
6: mixed contracts / predominant purpose
7: course of dealing
8: due diligence / notice / inspection / assumption of risk
9: disclaimer of warranty
10: revocation / recission / cancellation
11: express warranty
12: breach / notice
13: modification of warranty

14: unjust enrichment

15: fraud / deceit / coercion

16: actual damages / market value

17: mistake

18: statute of limitations

19: consideration

20: course of conduct / dealing

21: election of remedies / inconsistent remedy

22: apparent authority

23: statute of frauds / oral contracts

24: consequential damages / punitive damages

25: liquidated damages

26: economic loss rule

27: implied warranty of title

28: implied warranty of usage of trade

29: performance

30: offer

31: acceptance / contract formation

32: rejection / nonconformity

33: implied warranty of fitness for particular purpose

34: privity of contract / third-party liability / assignment

35: repudiation / anticipatory breach

36: right of adequate assurance

37: cure

38: specific performance

39: parol evidence / ambiguity

40: conflicting terms / battle of forms

41: unconscionability / illegality

42: risk of loss

43: cover / recover

44: resale

45: open terms / indefiniteness

46: entrustment / passing of title

47: contract modification / waiver

48: delivery / stoppage

49: remedy limitation

50: duty to mitigate

51: sale elements (generally)

52: UCC Definitions / Gap Fillers

B. cats_1.txt

This file contains a truncated version of the cats_0.txt. This file is enclosed within the submission and the values can be found within the data frame under the cats_0 column. Whereas cats_0.txt contains 53 total categories, cats_1 contains only 19. The categories are as follows:

0: other

1: promissory estoppel / detrimental reliance

2: Contract Interpretation / Form Requirements

3: Contract Formation

4. Implied Warranties

5. Mixed Contracts

6. Due Diligence (Notice / Nonconformity)

7. Disclaimer and Modification of Warranties

8. Contract Termination

9: Express Warranties

10: Breach / Repudiation

11: Remedies

12: Voidable Conditions

13: Remedy Limitations

14: Passage of Title / Risk of Loss

15. Contract Modification / Waiver

16: Performance

17 Privity of Contract

18: Loss Mitigation

2. Data Dictionary (Data Frame: filerpar2.csv)

The data frame contains 8 total columns. The columns are listed below with a description of the data.

- "Folder": Datatype –int. This is the location of the case within the directory that was used to compile the data frame. I kept this here, so that I could ensure the cases were properly stored inside the data frame.
- "file": Datatype—str. This is the file name of the row's case.
- "case_name": Datatype—str. This is the case name of the row's case.
- "date": Datatype—str. This is the date of the row's case. It is in the following format: [Month Proper Name] DD, YYYY.
- "maj_op": Datatype—str. This is a paragraph within a case.
- "wcontext": Datatype—str. This is the target paragraph found within "maj_op" and the surrounding paragraph on either side. If the paragraph is the first in the case, then only the following paragraph is included, and if the paragraph is the last

in the case, then only the preceding paragraph is included. Otherwise, normally this will be three paragraphs, with the center paragraph being the target paragraph.

- "cat_0": Datatype—int. This is a label for the row. Use cats_0.txt to decode.
- "cat_1": Datatype—int. This is a label for the row. Use cats_1.txt to decode.
- "UCC": Datatype—int, Boolean: This is a label row which represents a zero or non-zero value in the cat_0 column, with 1's representing a non-zero value.

3. Model Architecture

Fig. 1. Model 1 A, & B

Model: "functional_4"

Layer (type)	Output Shape	Param #	Connected to
attention_mask (InputLayer)	(None, 512)	0	-
input_ids (InputLayer)	(None, 512)	0	-
legal_bert_embeddi… (LegalBERTEmbeddin…	(None, 512, 768)	0	attention_mask[0… input_ids[0][0]
global_average_poo… (GlobalAveragePool…	(None, 768)	0	legal_bert_embed…
dense_8 (Dense)	(None, 128)	98,432	global_average_p…
dropout_4 (Dropout)	(None, 128)	0	dense_8[0][0]
dense_9 (Dense)	(None, 53)	6,837	dropout_4[0][0]

Total params: 105,269 (411.21 KB)
Trainable params: 105,269 (411.21 KB)
Non-trainable params: 0 (0.00 B)

Fig. 2. Model 1 C

Model: "functional"

Layer (type)	Output Shape	Param #	Connected to
attention_mask (InputLayer)	(None, 512)	0	-
input_ids (InputLayer)	(None, 512)	0	-
legal_bert_embeddi… (LegalBERTEmbeddin…	(None, 512, 768)	0	attention_mask[0… input_ids[0][0]
global_average_poo… (GlobalAveragePool…	(None, 768)	0	legal_bert_embed…
dense (Dense)	(None, 128)	98,432	global_average_p…
dropout (Dropout)	(None, 128)	0	dense[0][0]
dense_1 (Dense)	(None, 19)	2,451	dropout[0][0]

Total params: 100,883 (394.07 KB)
Trainable params: 100,883 (394.07 KB)
Non-trainable params: 0 (0.00 B)

4. Training Loss / Accuracy Visualizations

Fig. 3. Model 1 B. Accuracy and Loss Function Training Curves by Epoch (“wcontext”).

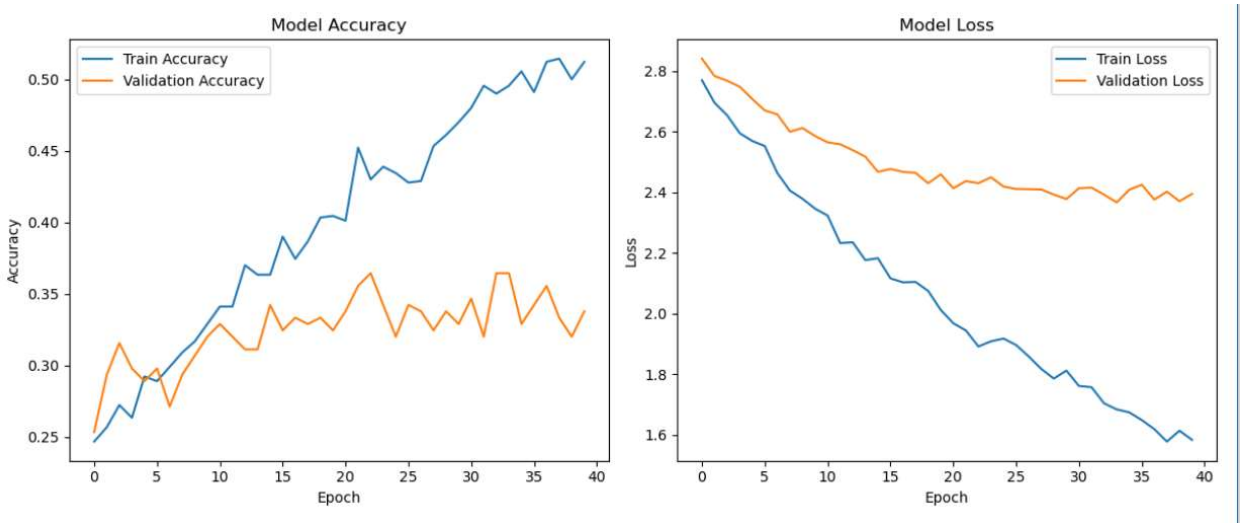


Fig. 4. Model 1 B. Accuracy and Loss Function Training Curves by Epoch ("maj_op")

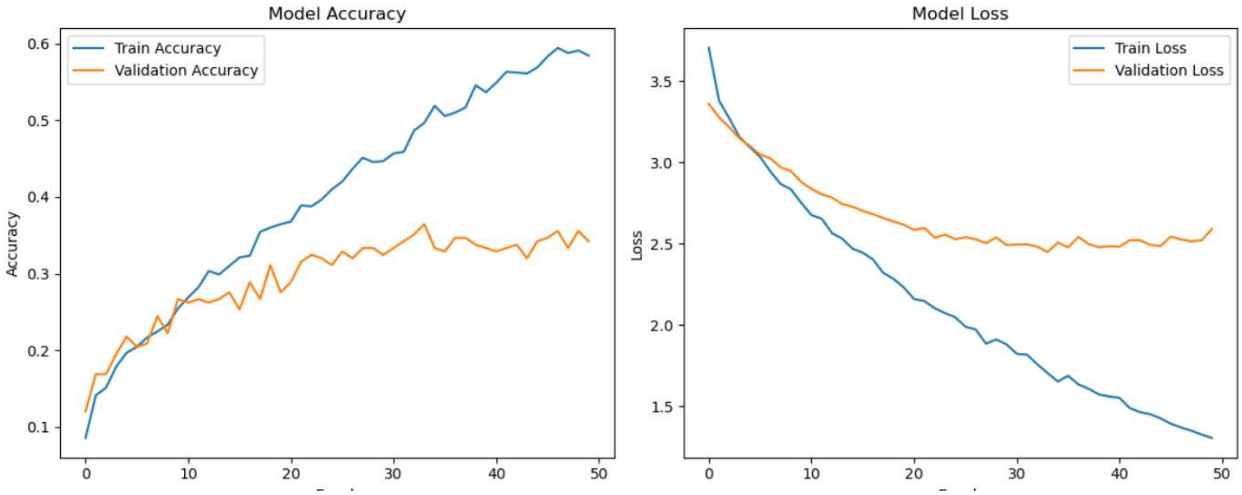


Fig. 5. Model 1 C. Accuracy and Loss Function Training Curves by Epoch (data = "maj_op," label = "cat_1," with category 0).

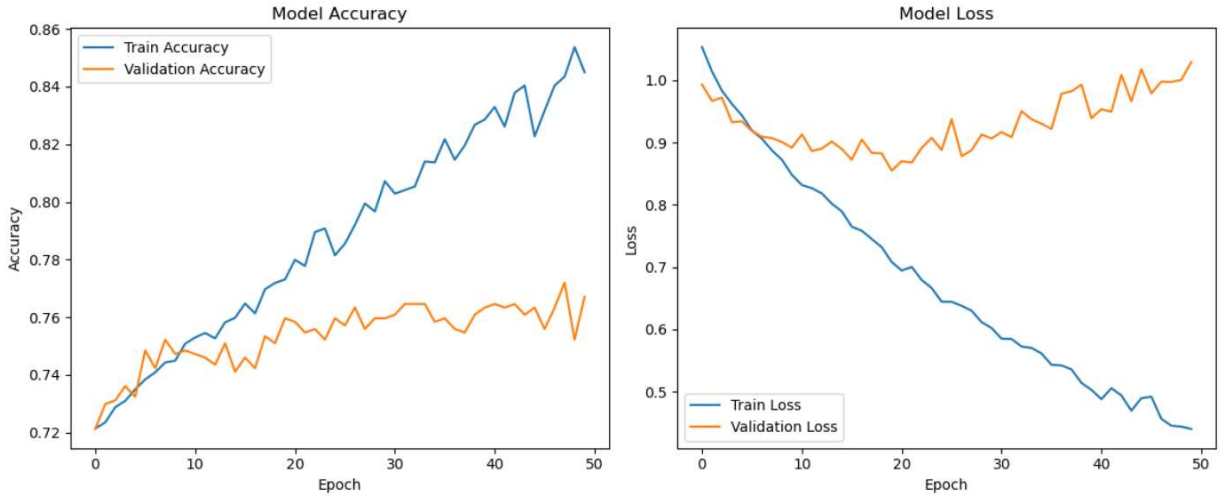
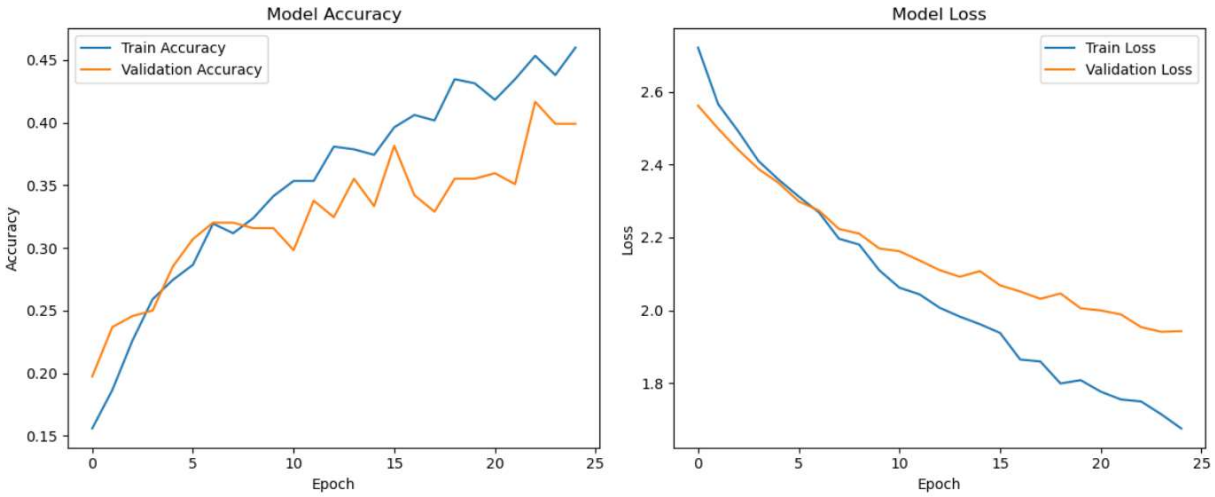


Fig. 6. Model 1 C. Accuracy and Loss Function Training Curves by Epoch

(data = "maj_op," label = "cat_1" without category 0



5. Performance Metric Reports

A. Multiclass BERT Models

BERT Model 1A (‘wcontext,” “cat_0” category 0 retained)

Accuracy: 0.7264618434093162

Precision (macro): 0.243331586196352

Recall (macro): 0.147725231692623

F1 Score (macro) 0.17202024041991176

	precision	recall	f1-score	support
0	0.78	0.95	0.86	720
1	0.00	0.00	0.00	2
2	0.00	0.00	0.00	3
3	0.00	0.00	0.00	3

4	0.00	0.00	0.00	4
5	0.80	0.31	0.44	13
7	0.00	0.00	0.00	1
8	0.00	0.00	0.00	14
9	0.00	0.00	0.00	1
10	0.00	0.00	0.00	7
11	0.31	0.42	0.36	12
12	0.27	0.09	0.14	33
13	0.50	0.33	0.40	3
14	0.25	0.17	0.20	6
15	0.56	0.31	0.40	16
16	0.29	0.15	0.20	27
17	0.00	0.00	0.00	2
18	0.57	0.50	0.53	8
19	0.00	0.00	0.00	13
20	0.00	0.00	0.00	1
21	0.00	0.00	0.00	2
22	1.00	0.50	0.67	2
23	0.70	0.35	0.47	20
24	0.00	0.00	0.00	4
25	0.00	0.00	0.00	1
27	1.00	1.00	1.00	1
29	1.00	0.20	0.33	5
30	0.00	0.00	0.00	1
31	0.00	0.00	0.00	9
32	0.00	0.00	0.00	3
33	0.00	0.00	0.00	1
34	0.33	0.25	0.29	16
35	1.00	0.25	0.40	4

36	0.00	0.00	0.00	1
37	0.00	0.00	0.00	1
39	0.00	0.00	0.00	6
41	0.00	0.00	0.00	3
42	0.00	0.00	0.00	1
43	0.00	0.00	0.00	3
46	0.50	0.50	0.50	2
47	0.50	0.17	0.25	6
48	0.33	0.10	0.15	10
49	0.00	0.00	0.00	7
50	0.00	0.00	0.00	1
51	0.00	0.00	0.00	2
52	0.50	0.25	0.33	8
accuracy			0.73	1009
macro avg	0.24	0.15	0.17	1009
weighted avg	0.64	0.73	0.67	1009

BERT Model 1B (“wcontext,” “cat_0” Category 0 Dropped)

Accuracy: 0.33687943262411346

Precision (macro): 0.3013950797588749

Recall (macro): 0.20528658333536381

F1 Score (macro) 0.2195490253577545

	precision	recall	f1-score	support
1	0.00	0.00	0.00	2
2	0.00	0.00	0.00	3
3	0.00	0.00	0.00	1

4	0.25	0.09	0.13	11
5	0.50	0.56	0.53	9
7	0.00	0.00	0.00	1
8	0.00	0.00	0.00	10
9	0.00	0.00	0.00	1
10	0.50	0.14	0.22	7
11	0.44	0.33	0.38	12
12	0.20	0.61	0.30	33
13	0.00	0.00	0.00	3
14	0.67	0.44	0.53	9
15	0.30	0.44	0.36	16
16	0.41	0.58	0.48	26
17	0.00	0.00	0.00	1
18	0.50	0.33	0.40	6
19	0.00	0.00	0.00	9
21	1.00	0.33	0.50	3
22	0.00	0.00	0.00	3
23	0.59	0.67	0.62	15
24	0.00	0.00	0.00	9
25	0.00	0.00	0.00	1
27	1.00	0.50	0.67	2
29	1.00	0.25	0.40	4
31	0.00	0.00	0.00	8
32	0.00	0.00	0.00	3
33	0.00	0.00	0.00	3
34	0.53	0.56	0.55	16
35	0.00	0.00	0.00	3
39	0.20	0.12	0.15	8
40	0.33	0.50	0.40	2

41	0.00	0.00	0.00	2
43	1.00	0.25	0.40	4
45	0.00	0.00	0.00	1
46	0.50	0.50	0.50	4
47	0.33	0.33	0.33	6
48	0.50	0.38	0.43	8
49	1.00	0.17	0.29	6
51	0.00	0.00	0.00	2
52	0.60	0.33	0.43	9
accuracy			0.34	282
macro avg	0.30	0.21	0.22	282
weighted avg	0.35	0.34	0.31	282

BERT Model 1B (“maj_op,” “cat_0” category zero dropped)

Accuracy: 0.3723404255319149

Precision (macro): 0.19708313819602072

Recall (macro): 0.1821879678586996

F1 Score (macro) 0.17121401884546608

	precision	recall	f1-score	support
1	0.00	0.00	0.00	2
2	0.00	0.00	0.00	3
3	0.00	0.00	0.00	1
4	0.40	0.18	0.25	11
5	0.43	0.67	0.52	9

7	0.00	0.00	0.00	1
8	0.33	0.20	0.25	10
9	0.00	0.00	0.00	1
10	0.00	0.00	0.00	7
11	0.27	0.25	0.26	12
12	0.33	0.70	0.45	33
13	0.00	0.00	0.00	3
14	0.58	0.78	0.67	9
15	0.63	0.75	0.69	16
16	0.36	0.69	0.47	26
17	0.00	0.00	0.00	1
18	0.00	0.00	0.00	6
19	0.31	0.44	0.36	9
21	0.00	0.00	0.00	3
22	0.00	0.00	0.00	3
23	0.40	0.40	0.40	15
24	0.50	0.22	0.31	9
25	0.00	0.00	0.00	1
27	1.00	0.50	0.67	2
29	0.00	0.00	0.00	4
31	0.20	0.12	0.15	8
32	0.00	0.00	0.00	3
33	0.00	0.00	0.00	3
34	0.35	0.69	0.47	16
35	0.00	0.00	0.00	3
39	0.60	0.38	0.46	8
40	0.00	0.00	0.00	2
41	0.00	0.00	0.00	2
43	0.00	0.00	0.00	4

45	0.00	0.00	0.00	1
46	0.00	0.00	0.00	4
47	0.00	0.00	0.00	6
48	0.00	0.00	0.00	8
49	1.00	0.17	0.29	6
51	0.00	0.00	0.00	2
52	0.38	0.33	0.35	9
accuracy			0.37	282
macro avg	0.20	0.18	0.17	282
weighted avg	0.31	0.37	0.31	282

BERT Model 1C ("maj_op," "cat_1" with category zero).

Accuracy: 0.7611496531219029

Precision (macro): 0.3419469805552352

Recall (macro): 0.20337336497762937

F1 Score (macro) 0.23170006675653987

	precision	recall	f1-score	support
0	0.83	0.97	0.89	724
1	0.00	0.00	0.00	2
2	0.36	0.44	0.40	39
3	0.50	0.12	0.20	24
4	0.43	0.12	0.18	26
6	0.75	0.21	0.33	14
7	0.00	0.00	0.00	4
8	0.25	0.10	0.14	10
9	0.11	0.08	0.10	12
10	0.32	0.30	0.31	33
11	0.41	0.26	0.32	46
12	0.69	0.58	0.63	19

13	0.50	0.31	0.38	13
14	0.00	0.00	0.00	13
15	1.00	0.17	0.29	6
16	0.00	0.00	0.00	4
17	0.00	0.00	0.00	16
18	0.00	0.00	0.00	4
accuracy			0.76	1009
macro avg	0.34	0.20	0.23	1009
weighted avg	0.70	0.76	0.72	1009

BERT Model 1C (maj_op without 0)

Accuracy: 0.36140350877192984

Precision (macro): 0.27540398023832774

Recall (macro): 0.20363696798109235

F1 Score (macro) 0.20027323314961756

	precision	recall	f1-score	support
1	0.00	0.00	0.00	2
2	0.38	0.79	0.52	39
3	0.21	0.12	0.16	24
4	0.38	0.42	0.40	26
6	0.20	0.07	0.11	14
7	0.00	0.00	0.00	4
8	0.00	0.00	0.00	10
9	1.00	0.17	0.29	12
10	0.25	0.48	0.33	33
11	0.40	0.57	0.47	46
12	0.50	0.21	0.30	19
13	0.00	0.00	0.00	13
14	0.80	0.31	0.44	13

15	0.00	0.00	0.00	6
16	0.00	0.00	0.00	4
17	0.56	0.31	0.40	16
18	0.00	0.00	0.00	4
accuracy			0.36	285
macro avg	0.28	0.20	0.20	285
weighted avg	0.35	0.36	0.31	285

B. Binary Classification Models

Feed Forward Sequential Neural Network

Precision	0.820696
Recall	0.833365
Accuracy	0.855302
F1 score	0.826518

BERT

Precision	0.857143
Recall	0.501730
Accuracy	0.714569
F1 score	0.826518

LSTM

Accuracy: 0.8325074331020813

Precision: 0.7086330935251799

Recall: 0.6912280701754386

F1 Score: 0.6998223801065719

Classification Report:

	precision	recall	f1-score	support
0	0.88	0.89	0.88	724
1	0.71	0.69	0.70	285
accuracy			0.83	1009
macro avg	0.79	0.79	0.79	1009
weighted avg	0.83	0.83	0.83	1009

Confusion Matrix:

[[643 81]
[88 197]]