Authorship Verification of the Disputed Pauline Letters through Deep Learning

Evy Beijen
Vrije Universiteit, Amsterdam
e.beijen@vu.nl

Rianne de Heide
University of Twente, Enschede
r.deheide@utwente.nl

Abstract: In the Christian tradition, fourteen letters of the New Testament have been attributed to the Apostle Paul. However, for seven of these letters—1 and 2 Timothy, Titus, Ephesians, Colossians, 2 Thessalonians, and Hebrews —the attribution to Paul has been the subject of scholarly debate. This study aims to develop a bidirectional long short-term memory (BiLSTM) network to classify chunks of these disputed letters, each approximately 100 words long, as either authored by Paul or not. Two model variants—a plaintext variant and a lemmatized text variant—were trained on undisputed Pauline letters and 'impostor letters' which serve as negative examples of Paul's writing. The plaintext variant achieved 84% accuracy and the lemmatized text variant 83% accuracy. Both variants classify the majority of text chunks from Colossians and 2 Thessalonians as Pauline and the majority of chunks from Hebrews and 1 Timothy as non-Pauline, although caution is warranted in drawing strong conclusions from these results. For the remaining disputed Pauline letters—Titus, 2 Timothy, and Ephesians—the majority classification varies between the model variants, further emphasizing the need for caution. Nevertheless, this study introduces a deep learning approach to the authorship verification problem of the disputed Pauline letters, potentially serving as a model for future research.

Keywords: Pauline epistles, authorship attribution, deep learning, text classification

1. Introduction

In the Christian tradition, fourteen letters of the New Testament have been attributed to the Apostle Paul (Ehrman 2020). These letters, written in the first or second century CE in Koine Greek, played an important role in shaping early Christian theology and ethics. However, for seven of these letters—Titus, 1 and 2 Timothy, Hebrews, Ephesians, Colossians, and 2 Thessalonians—the attribution to Paul has been the subject of scholarly debate. Arguments regarding their authorship include considerations of writing style, theological content, and historical context, among other factors (Ehrman 2020). The attribution debate surrounding these texts forms the focus of this article, in which deep learning methods are employed to investigate the authorship of the disputed Pauline letters.[1]

[1] In current scholarly discussion, Hebrews is rarely classified as a (disputed) Pauline letter. Whereas Titus, 1 and 2 Timothy, Ephesians, Colossians, and 2 Thessalonians all explicitly claim to be written by Paul, Hebrews is anonymous. Its

The disputed Pauline letters present us with an "open set authorship attribution" problem, also referred to as "authorship verification" (Sari 2018, 17–20). In "closed set authorship attribution," a complete list of possible authors is available, each with a corpus of written work for comparison. That is, the true author is definitely included in the list. By contrast, in authorship verification—such as with the disputed Pauline letters—no such definitive list exists. This distinction makes authorship verification more complex than closed set authorship attribution, but various computational methods have been developed to address this challenge, which can be divided into intrinsic and extrinsic methods (Sari 2018, 19).

Contrasting extrinsic methods, intrinsic methods do not use any additional documents to aid the verification process. That is, intrinsic methods only use the documents definitively written by a particular author and documents possibly written by that same author, for which authorship is to be verified. Extrinsic methods, on the other hand, approach an authorship verification problem by transforming it into a binary or multiclass classification task, by using a set of "impostor documents" (Sari 2018, 20). These impostor documents are texts not written by the suspected author and are used to establish a baseline for comparison. To clarify, in extrinsic methods, a document is classified as written by a particular author if it is more similar to documents known to be written by that author than to the impostor documents. As noted by Yunita Sari (2018, 20), various studies have found superior performance of extrinsic methods compared to intrinsic methods, which informed our decision to adopt an extrinsic approach for this study.

In recent history, several computational approaches have been applied to the authorship verification problem of the disputed Pauline letters. An early example is A.Q. Morton's work (1965; 1971), which used word frequencies, sentence length, and elementary statistics to investigate the authorship of the letters attributed to Paul. Morton's work has been criticized in later studies, including by David L. Mealand (1989), who was among the first to apply multivariate statistical methods to study the (disputed) Pauline letters (Mealand 1995). He used a variety of linguistic features, such as part-of-speech tag frequencies and most frequent words, to determine which New Testament letters are 'nearest' to each other. More recently, Jacques Savoy (2019) published a study on the authorship of the (disputed) Pauline letters using two computational approaches. The first approach used Burrows' Delta and Labbé's intertextual distance—both based on word frequencies—to measure similarity between texts, followed by hierarchical clustering. The second approach employed an extrinsic method based on the most frequent *n*-grams, using non-Pauline New Testament letters as impostor documents. The identified clusters—{Romans, Galatians, 1 and 2 Corinthians}, {1 and 2 Thessalonians}, {Colossians, Ephesians}, and {Titus, 1 and 2 Timothy}—largely align with Mealand's findings. However, Savoy's second approach revealed connections between letters from different clusters, complicating straightforward interpretations.

association with Paul emerged as Christians in the third and fourth centuries became convinced the letter was authored by him (Ehrman 2020, 481). Although modern scholarship generally agrees that Paul did not write this letter, Hebrews was included in this study due to its traditional association with Paul and its potential value for the computational analysis. Importantly, the methodology used in this study does not make any assumptions about the authorship of Hebrews.

Other recent computational studies on the letters attributed to Paul used hand-coded stylistic features, term frequency-inverse document frequency, and cosine similarity (Roy and Robertson 2023; Van der Ventel and Newman 2022). However, more advanced computational techniques appear underexplored in the context of the disputed Pauline letters. Specifically, to our knowledge, there have been no studies applying deep learning techniques to investigate the authorship of the disputed Pauline letters, despite the frequent use of deep learning in authorship attribution tasks, where it is valued for its ability to analyze sequential data and learn complex patterns (Sari 2018, 11, 53). This study aims to address this gap by introducing a deep learning approach to the authorship verification problem of the disputed Pauline letters, potentially providing a model for future research in this area.

Jordan Perry's research (2021) on Plato's disputed Epistle VII is an example of deep learning techniques being applied to a similar authorship verification problem as the disputed Pauline letters. Perry developed a bidirectional long short-term memory (BiLSTM) network with trainable word embeddings to classify segments of roughly 100 words of Ancient Greek text as written by Plato or by one of six other Ancient Greek authors. The writings of these six authors serve as impostor documents, i.e., as negative examples of Plato's writing. Inspired by Perry's work, we decided to adopt a similar approach for our research. That is, this study aims to develop a BiLSTM network with a trainable embedding layer to classify chunks of the disputed Pauline letters, each roughly 100 words long, as either authored by Paul or not.

The paper is structured as follows. In Section 2, we introduce the machine learning methods used in this study and the technical terms in a way accessible to those who have not used or heard of these methods before. We explain our methodology in Section 3: our assumptions, our data and its preprocessing, a brief summary of the model architecture, and a description of how we train, tune, and test the model. In Section 4, we present the results, and we end with a discussion and a conclusion in Section 5.

2. Technical preliminaries

In this study, we make use of a long short-term memory (LSTM) network, which is a specialized variant of a recurrent neural network (RNN). We first explain the basics of RNNs in Section 2.1 before introducing the specifics of an LSTM network in Section 2.2. Lastly, in Section 2.3, we discuss the method we use for hyperparameter tuning, namely Bayesian optimization.

2.1 Recurrent neural networks

RNNs are deep learning models that have found extensive application in various domains, particularly handling sequential data like text or audio. These models usually consist of standard recurrent cells (Yu et al. 2019, 1237). In a standard recurrent cell, information flows from one step to the next through a 'hidden state'. This hidden state acts as a memory, retaining information about the sequence (in this case: text) processed thus far. At each step, the recurrent cell takes an input (in this case: a word), combines it with the previous hidden state, and computes a new hidden state. RNNs often consist of multiple recurrent cells in a certain structured design and configuration (Yu et al. 2019, 1236). This configuration is called the network's architecture, which often consists of several layers.

The input layer serves as the entry point for the data. Here, we use an embedding layer to embed the words in the texts into vector representations. The hidden layers are responsible for processing and transforming the data. The output layer is responsible for producing the final output, such as a prediction that the text is written by a certain author.

In each cell, the process from input to output happens through a function that has a specific form—a classical example is the sigmoid function—and has several weights by which the input is multiplied. For different problems, different weights are appropriate, and these are learned by the model using training data: data from which we know the true label (in our case: written by Paul, or written by someone else). We specify a loss function that quantifies the discrepancy between the predicted outputs of the model and the target values of the output. We then 'try out' different configurations of the weights and we choose the model (i.e., a set of weights) for which our loss function is minimized on the training data; this is called 'training the model'. Because there are far more possible configurations of weights than we will ever be able to compute, we use smart ways for training, often a form of gradient descent. There are many popular and standard packages one can use for such optimization tasks, and we will detail in Section 3.3 which one we employ.

2.2 Long short-term memory (LSTM) networks

While standard recurrent cells have shown success in certain tasks, they have often been found lacking in capturing long-term dependencies (Yu et al. 2019, 1236). To deal with this problem, Hochreiter and Schmidhuber (1997) proposed the LSTM cell. LSTM networks use these specialized recurrent cells, which typically feature three types of gates: the input gate, the forget gate, and the output gate. These components work together to control the flow of information into, out of, and within the recurrent cells. This gating mechanism allows LSTM networks to retain important information over longer data sequences and 'forget' irrelevant details, thereby enabling them to effectively model complex sequential patterns. This makes them particularly effective for language modeling.

2.3 Bayesian optimization

There are a number of hyperparameters that must be specified before training the model, such as the hidden layer size, the embedding dimension, and more. Different approaches are possible to find suitable hyperparameter values, such as grid search or Bayesian optimization. Bayesian optimization (Snoek, Larochelle, and Adams 2012) is an approach suited to optimizing functions that are expensive to evaluate, as is the case for the performance of the LSTM network used in this study. Generally, Bayesian optimization is an iterative approach to search for the values of the hyperparameters that maximize some objective function. This objective function evaluates the model's performance corresponding to specific hyperparameter values. The distinctive aspect of Bayesian optimization compared to other optimization procedures is its construction of a probabilistic model for the objective function. Instead of exhaustively exploring the entire search space, Bayesian optimization iteratively constructs a probabilistic model that guides the selection of the next configuration of the hyperparameters to evaluate.

3. Methodology

3.1 Assumptions

To set up our methodology, we first made several assumptions about the authorship of the letters in our dataset. Given that the majority of New Testament scholars regard these letters as authentically Pauline (Ehrman 2020, 341), we assume that Paul authored the following seven: Romans, 1 and 2 Corinthians, Galatians, Philippians, 1 Thessalonians, and Philemon. We will refer to these letters as the undisputed Pauline letters. In addition, we assume that the following seven letters, which we refer to as the disputed Pauline letters, were possibly but not certainly written by Paul: Titus, 1 and 2 Timothy, Ephesians, Colossians, 2 Thessalonians, and Hebrews. Hence, we want our model to predict for these letters whether or not Paul is the author.

Furthermore, we selected several 'impostor documents' for Paul's writings. As Koppel and Winter (2014) emphasize, the performance of extrinsic methods for authorship verification tasks is highly dependent on the quality (i.e., similarity to the 'suspect' texts) and quantity of the impostor documents. They found that extrinsic methods do not perform well when the genre and/or topics of the impostor documents are different from the other input documents. Therefore, we chose impostor letters written in the same language, in approximately the same time period, in the same genre, and about similar topics as the (un)disputed Pauline letters, ensuring a comparable dataset. The selected impostor letters are James, 1 and 2 Peter, 1, 2, 3 John, Jude, 1 Clement, the Epistle of Barnabas, and the seven letters of Ignatius of Antioch. We are thus assuming that none of these letters were written by Paul, as existing research gives us no reason to assume otherwise.

Additionally, we align with current scholarship on early Christianity, as presented by Bart D. Ehrman (2020), regarding the authorship of certain impostor letters. That is, we assume that 1, 2, and 3 John were written by the same author and that the seven letters of Ignatius of Antioch were written by the same author, but we assume that 1 and 2 Peter were written by different authors.

3.2 Dataset and preprocessing

The Greek texts of the (un)disputed Pauline letters and impostor letters were extracted from the websites of the *Society of Biblical Literature* (SBL) and the *Christian Classics Ethereal Library* (SBL 2023; Lake 1913). We loaded the files containing these texts into Python. Then, this dataset was split into subsets for different purposes, as shown in Table 1. We decided to exclude two undisputed Pauline letters and two impostor letters from Subset A to create Subset B. The purpose of Subset B was to enable assessment of the model's performance on entirely unseen letters at the end, i.e., on letters of which no chunks were included in the training data, as an additional check of the model's effectiveness.

Table 1. Purpose, included letters, and number of chunks per subset

Subset	Purpose	Included letters	#chunks
A	Training and testing	**Undisputed**: Romans, 1 and 2 Corinthians, Philippians, 1 Thessalonians **Impostors**: James, 2 Peter, 1, 2, 3 John, 1 Clement, the Epistle of Barnabas, the seven letters of Ignatius of Antioch	549
B	Assessing performance on entirely unseen letters (with known authorship)	**Undisputed**: Philemon, Galatians **Impostors**: Jude, 1 Peter	54
C	Predicting authorship	**Disputed**: Titus, 1 and 2 Timothy, Ephesians, Colossians, 2 Thessalonians, Hebrews	157

In selecting the two undisputed Pauline letters for Subset B, our aim was to minimize data loss for Subset A. We chose Philemon for Subset B because it is Paul's shortest letter, thereby preserving the maximum amount of material for Subset A. Additionally, Galatians was selected for Subset B as a letter of moderate length due to its stylistic similarity, as indicated by previous research (Savoy 2019, 1095), with longer letters: Romans, 1 Corinthians, and 2 Corinthians. This decision was made in order to preserve as much of the stylistic diversity of Subset A as possible.

For the two impostor letters of Subset B, our aim was to include potentially challenging letters for the model to classify correctly, while still preserving sufficient data for Subset A. We specifically chose letters whose authors likely did not write any other letters in the dataset, according to current scholarship regarding their authorship. Such letters are likely more challenging for the model compared to letters by authors with multiple letters in the impostor documents (and thus in the training data). Consequently, we chose Jude, the shortest letter by a unique author, and 1 Peter, a letter of moderate length by a unique author. Notably, 1 Peter exhibits stylistic similarities with Philippians according to Savoy's research (2019, 1092), one of the Pauline letters in Subset A, potentially making it challenging for the model to recognize it as non-Pauline.

In further preprocessing, all letters were split into chunks of approximately 100 words. This splitting process involved splitting the letters into sentences, followed by concatenating consecutive sentences into chunks, ensuring that each chunk adhered to the 100-word limit. Additionally, punctuation, accentuation, and capitalization were removed from the texts, since no consistent punctuation, accentuation, or capitalization was found in the earliest manuscripts of the letters (Metzger and Ehrman 2005). Furthermore, as some authors stated their names in their letters, these names were replaced by the token <n> to prevent the model from learning to classify texts based on the name stated by the author.

Lemmatization was performed on all text chunks using a pre-trained lemmatizer from the Classical Language Toolkit (Johnson et al. 2021). Lemmatization means reducing words to their dictionary

form, causing different inflected forms of the same word to be treated as the same, whereas they are treated as distinct when using the plaintext version of the text. Lemmatization might help in authorship verification as it minimizes variability caused by different forms of the same word, placing more focus on the core vocabulary used by the author. For each subset, both versions of the text chunks were stored, i.e., the plaintext version and the lemmatized version. Then, to prepare the text chunks for being processed by the model, the chunks were tokenized into sequences of numbers and these sequences were padded to ensure uniform length, using the TensorFlow library (Abadi et al. 2015).

After these preprocessing steps, the plaintext and lemmatized versions of Subset A, intended for training and testing, each contained 549 tokenized chunks of which 45% were written by Paul and 55% were not. Each text chunk in Subset A was labeled either "Paul" or "Other", making the model's task into a binary classification task. Both the plaintext and lemmatized versions of Subset A were randomly split into a training set (80%) and test set (20%).

3.3 Model architecture

The model used in this study has three main components: a trainable embedding layer, a BiLSTM layer, and a dense output layer. As indicated in the previous section, the input sequences of the model are tokenized text chunks. These sequences are first transformed by the embedding layer into sequences of vectors, which are then relayed to the BiLSTM layer, consisting of LSTM cells. The bidirectional configuration of this layer, introduced by Graves and Schmidhuber (2005), processes the input sequences in both forward and backward directions, capturing context and dependencies in the text in 'hidden states'. The two final hidden states, i.e., from both directions, are concatenated and passed to a dense output layer with two units, corresponding to the target classes "Paul" and "Other". The softmax activation function is applied to generate a probability distribution over the target classes, with the highest probability determining the predicted class label for each input sequence.

The model was trained using mini-batch gradient descent, which means the model's weights were iteratively updated after processing mini-batches of training data. The sparse categorical cross-entropy loss function and the Adam optimizer (Kingma and Ba 2017) were used. Furthermore, dropout (Srivastava 2014) was applied in training the model. Dropout is a regularization technique that is widely used to reduce overfitting in neural networks. In the present BiLSTM model, dropout was applied to the input vectors of the BiLSTM layer. This can help prevent the BiLSTM network from memorizing the training data too closely, potentially leading to better generalization to new sequences. The model architecture, including the training procedure, was implemented in Python using Keras (Chollet et al. 2015).

3.4 Tuning, training, and testing

Before training and testing the final model, the model's hyperparameters were tuned through Bayesian optimization (Snoek, Larochelle, and Adams 2012), separately for the plaintext and lemmatized model variants. This process involved defining a Python function to be optimized (i.e., the objective function), which took values for each hyperparameter, constructed the BiLSTM model, conducted five-fold cross-validation on the training set (plaintext or lemmatized text version), and returned the average accuracy score. The ranges for the six hyperparameters we tuned, shown in Table 2, are

primarily based on commonly used values in similar models and computation time considerations. During the Bayesian optimization process, the hyperparameter values passed to the objective function could, in principle, take any value within the specified intervals. However, for discrete hyperparameters, the function cast the passed values to integers.

Table 2. Ranges per hyperparameter used in Bayesian optimization

Hyperparameter	**Range**
Embedding dimension	[50, 200]
Hidden layer size	[50, 200]
Dropout	[0.1, 0.5]
Learning rate	[0.001, 0.01]
Batch size	[32, 128]
Epochs	[10, 20]

Bayesian optimization was executed using the BayesOpt library (Martinez-Cantin 2014), starting with evaluating 10 initial points randomly sampled from the hyperparameter space defined by the ranges in Table 2. Following this initial exploration, the process continued with 30 more iterations. In each iteration, the next point to evaluate was selected through a Bayesian optimization process, employing the upper confidence bound (UCB) acquisition function with default value $\kappa = 2.576$. The decision to use 10 initial points and 30 optimization iterations was based on considerations regarding computation time, available time and resources, and the objective of achieving decent model performance (see Section 4.1).

After obtaining tuned hyperparameter values for both the plaintext and lemmatized text variants of the model, we trained each variant on its respective training set. Subsequently, we evaluated the performance of the model variants by testing them on their respective test sets. Next, we used the trained model variants to classify the text chunks in Subset B, ensuring that each variant processed the data in its appropriate form (plaintext or lemmatized text). Finally, we used the model variants to classify the chunks of Subset C, i.e., the disputed Pauline letters, again ensuring that each model variant processed the data in its appropriate form (plaintext or lemmatized text).

4. Results

4.1 Subset A

Subset A was used for hyperparameter tuning, training, and testing the model. Employing Bayesian optimization with 40 iterations in total, comprising 10 initial exploratory iterations and 30 optimization iterations, led to the tuned hyperparameter values shown in Table 3. For these values, the objective function returned average accuracy scores of 0.86 for the plaintext variant and 0.87 for the

lemmatized text variant. Based on these scores, indicating decent model performance, and considering practical factors such as computation time and available time and resources, we decided to continue with the obtained hyperparameter values from Table 3, rather than increasing the number of iterations or trying different settings (e.g., different κ values or hyperparameter ranges).

Table 3. Tuned hyperparameter values for each model variant

Hyperparameter	**Plaintext**	**Lemmatized text**
Embedding dimension	70	61
Hidden layer size	115	80
Dropout	0.10339881584988651	0.39510690225741535
Learning rate	0.001070605093208486	0.001
Batch size	73	50
Epochs	18	20

Table 4. Accuracy scores of the model variants on their respective training and test sets

	Plaintext	**Lemmatized text**
Accuracy on training set	1.00	1.00
Accuracy on test set	0.84	0.83

Training the model variants with the hyperparameter values from Table 3 on their respective training sets and testing them on their respective test sets resulted in the accuracy scores given in Table 4. The scores show that the model variants achieve similar accuracy levels. We note that the accuracy scores on the test sets are lower than the average accuracy scores achieved through cross-validation in the Bayesian optimization process. These differences could be due to the specific training-test splits, especially considering the limited size of the dataset. Furthermore, for both model variants, there is a notable difference between the accuracy score on the test set and the training set, which is a common indication of overfitting. Overfitting occurs when a model fits too closely to the training data, resulting in poor generalization to unseen data. Furthermore, the confusion matrices in Figure 1 illustrate that both model variants produced more false negatives than false positives, with the lemmatized text variant showing a more pronounced imbalance.

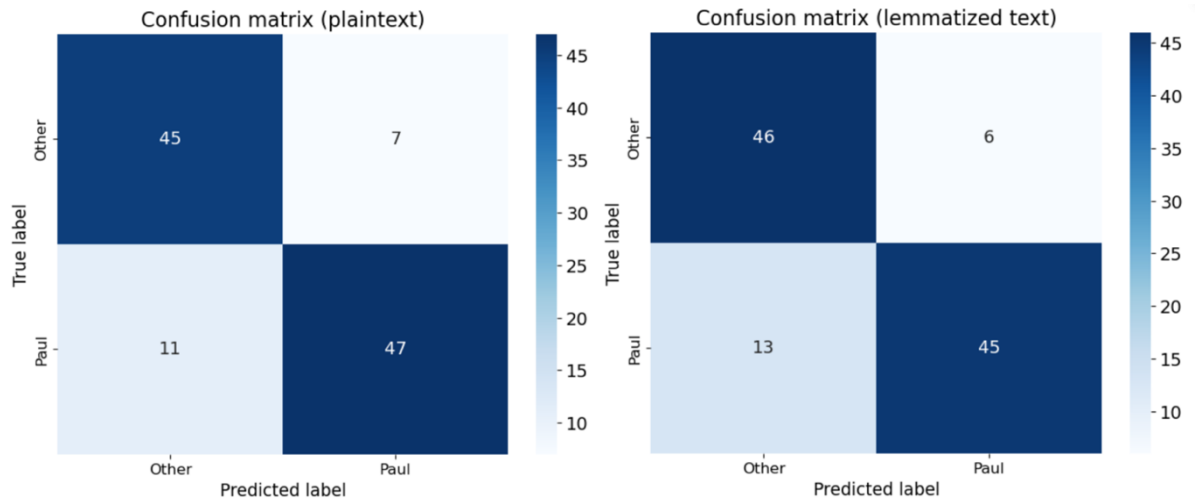

Figure 1. Confusion matrices from testing the model variants on their respective test sets

4.2 Subset B

Both model variants were used to classify the text chunks of Subset B in order to assess their performance on entirely unseen letters, i.e., letters of which no chunks were included in the training data. The results are given in Table 5. The model variants show similar numbers for the letters, although the plaintext variant performs slightly better on this subset. We observe that, despite some misclassifications, both models generally identify chunks from Galatians and Philemon as Pauline. However, they struggle more with recognizing the impostor letters as non-Pauline, especially 1 Peter. This suggests that the model's performance on completely unseen letters may be worse compared to the obtained accuracy scores.

Table 5. Predicted classes for completely unseen letters with known authorship (Subset B)

Letter	**True class**	**Plaintext (#chunks)**		**Lemmatized text (#chunks)**	
		Paul	Other	Paul	Other
Galatians	Paul	20 (80%)	5 (20%)	19 (76%)	6 (24%)
Philemon	Paul	3 (75%)	1 (25%)	3 (75%)	1 (25%)
1 Peter	Other	8 (42%)	11 (58%)	10 (53%)	9 (47%)
Jude	Other	2 (33%)	4 (67%)	2 (33%)	4 (67%)

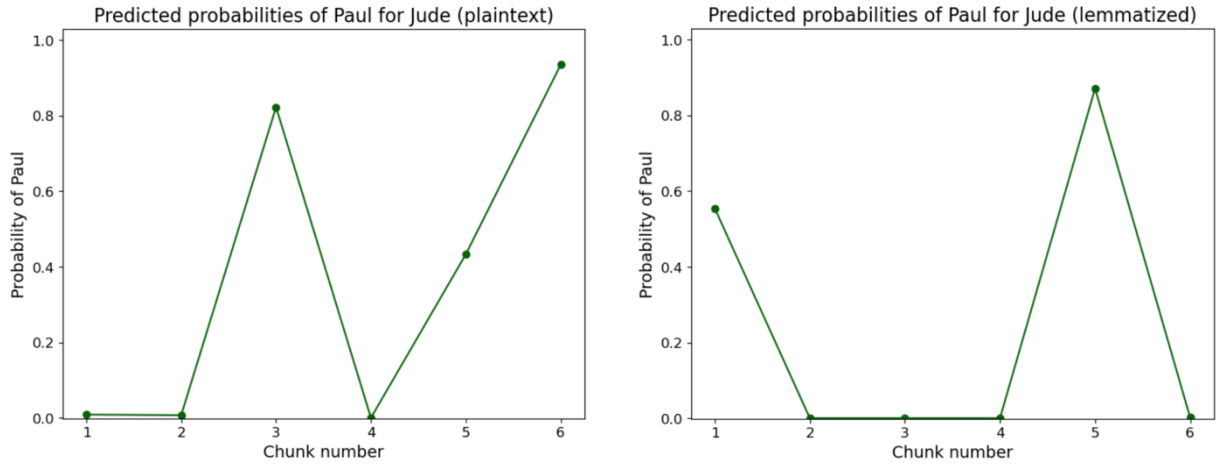

Figure 2. Predicted probabilities of being written by Paul for each chunk of Jude, by each model variant

Additionally, we extracted the model's predicted probabilities of text chunks being Pauline for each letter in Subset B and visualized them. Notably, for each of the four letters, the probability distribution is rather wide, with chunk probabilities ranging from nearly 0 to nearly 1. Figure 2 shows the plotted probabilities for Jude per model variant. Interestingly, while both variants classify the same number of chunks from Jude as Pauline and non-Pauline (see Table 5), there are differences in which particular chunks are categorized as such. Specifically, the model variants only agree on the classification of chunks 2 and 4 of Jude. By contrast, the model variants show complete agreement on the chunks of Philemon (see Figure 3). The patterns observed in the graphs for Galatians and 1 Peter show a mixture of overlapping and deviating characteristics between the model variants.

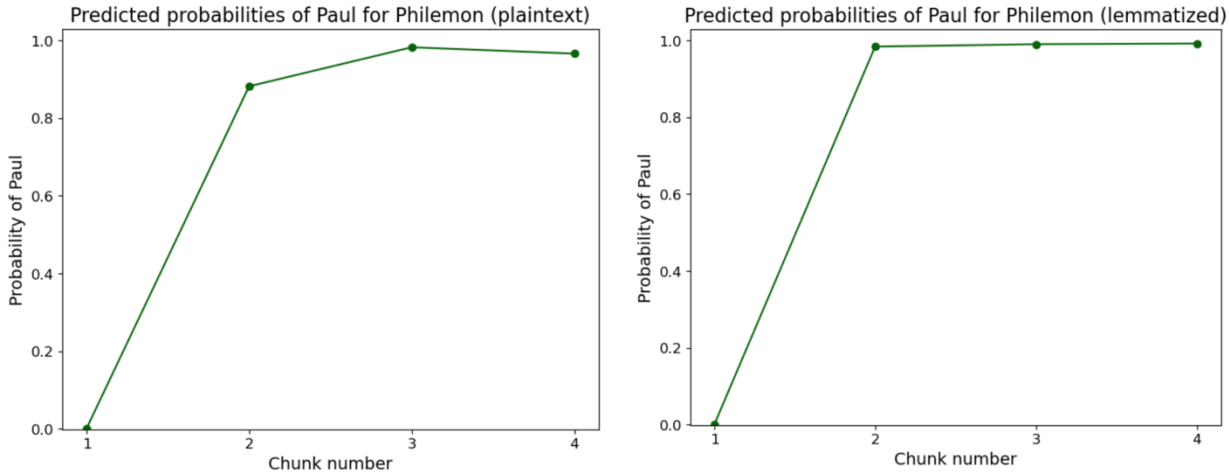

Figure 3. Predicted probabilities of being written by Paul for each chunk of Philemon, by each model variant

4.3 Subset C

Furthermore, both model variants were used to classify the text chunks of letters in Subset C, i.e., the disputed Pauline letters. Table 6 shows the predicted classes for the chunks for each model variant, with the majority classes highlighted in bold. Both model variants classify the majority of the text chunks from Colossians and 2 Thessalonians as Pauline, and the majority of the chunks from 1 Timothy and Hebrews as non-Pauline. Differences are observed, however, in the majority classification for other letters. A closer inspection of the numbers reveals that the lemmatized text variant consistently identifies an equal or greater number of chunks as written by Paul compared to the plaintext version.

Table 6. Predicted classes for the disputed Pauline letters (Subset C)

Letter	**Plaintext (#chunks)**		**Lemmatized text (#chunks)**	
	Paul	Other	Paul	Other
1 Timothy	6 (33%)	**12 (67%)**	6 (33%)	**12 (67%)**
2 Timothy	6 (40%)	**9 (60%)**	**8 (53%)**	7 (47%)
Titus	3 (38%)	**5 (62%)**	4 (50%)	4 (50%)
Hebrews	16 (28%)	**41 (72%)**	16 (28%)	**41 (72%)**
Ephesians	14 (50%)	14 (50%)	**15 (54%)**	13 (46%)
Colossians	**13 (62%)**	8 (38%)	**17 (81%)**	4 (19%)
2 Thessalonians	**7 (70%)**	3 (30%)	**8 (80%)**	2 (20%)

As before, we extracted and visualized the predicted probabilities for each letter in Subset C. As with Subset B, each letter in Subset C also exhibits a wide probability distribution, with chunk probabilities ranging from nearly 0 to nearly 1. Figure 4 shows the predicted probabilities for 2 Thessalonians per model variant. The patterns in both graphs largely align, except for a deviation at chunk 2. In graphs for the remaining six disputed Pauline letters, the plaintext and lemmatized text variants show differences for more than one chunk. Additionally, the graphs show many large fluctuations within the letters, complicating straightforward interpretations.

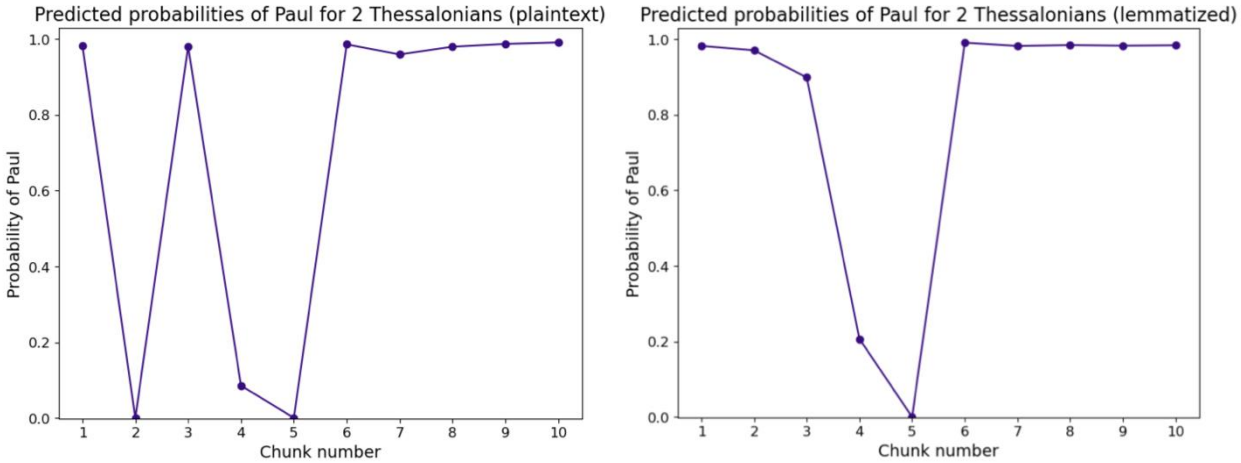

Figure 4. Predicted probabilities of being written by Paul for each chunk of 2 Thessalonians, by each model variant

5. Discussion and conclusion

Model performance

In this study, a BiLSTM model was developed to predict whether text chunks from the disputed Pauline letters were written by Paul. The two model variants—the plaintext variant and the lemmatized text variant—achieved accuracy scores of 0.84 and 0.83, respectively, indicating that the lemmatization process did not significantly impact overall model performance. However, lemmatization did influence the model's predictions. That is, further analysis of the predicted probabilities of text chunks being Pauline revealed many differences between the predictions of the two variants.

Critical remarks can be made about the model's performance. Firstly, both model variants show signs of overfitting in the disparity between accuracy scores on the training and test sets, despite attempts to mitigate this problem through dropout regularization. This highlights the model variants' limitations in their applicability beyond the training data. Moreover, we found indications of subpar performance on entirely unseen letters, i.e., letters of which no chunks were included in the training data. While both model variants performed well in identifying chunks from Galatians and Philemon as Pauline, both had particular difficulty with recognizing 1 Peter as non-Pauline. A possible explanation can be found in the previously discovered stylistic similarities between 1 Peter and Philippians (Savoy 2019, 1092), the latter being an undisputed Pauline letter included in the training data. This may have complicated the model's task of distinguishing 1 Peter as non-Pauline. However, even with this possible explanation, the model's difficulty in correctly classifying chunks from 1 Peter, combined with the wide range of probabilities observed for each tested letter, calls for caution in interpreting its predictions regarding the disputed Pauline letters.

Authorship of the disputed Pauline letters

Regarding the authorship of the disputed Pauline letters, both model variants classify the majority of the text chunks from Colossians and 2 Thessalonians as Pauline, and the majority of the chunks from

1 Timothy and Hebrews as non-Pauline. For the remaining three disputed Pauline letters—Titus, 2 Timothy, and Ephesians—the majority classification varies between the model variants. Interestingly, for all disputed letters, the lemmatized variant identifies an equal or greater number of chunks as Pauline compared to the plaintext variant. This observation suggests that focusing more on the core vocabulary used by the author, as done in the lemmatized variant, may result in a higher likelihood of attributing the disputed Pauline letters to Paul.

The classification of Hebrews as non-Pauline by both model variants aligns with current New Testament scholarship. Although the letter is anonymous, it was attributed to Paul as Christians in the third and fourth centuries became convinced of his authorship (Ehrman 2020, 481). Modern scholars, however, largely agree that Paul did not write Hebrews. Additionally, 1 Timothy is widely regarded as non-Pauline by scholars, which the model's predictions are in alignment with. In particular, most New Testament scholars agree that the Pastoral epistles—1 Timothy, 2 Timothy, and Titus—were written by the same author, who was not Paul (Ehrman 2020, 456–461). For these three letters, the plaintext variant classified the majority of text chunks as non-Pauline, although the margins were smaller for Titus (62%) and 2 Timothy (60%) compared to 1 Timothy (67%). The lemmatized variant diverges from the dominant scholarly opinion, as it does not classify the majority of text chunks from 2 Timothy or Titus as non-Pauline.

About Ephesians, Colossians, and 2 Thessalonians, there is less scholarly consensus. As such, it is not possible to make definitive statements about whether the predictions of the model variants align with current scholarship. While both variants classify the majority of text chunks from Colossians and 2 Thessalonians as Pauline, suggesting these letters are likely authentic based on our results, we should exercise caution in drawing conclusions about the authorship of any of the disputed letters, given the model's shortcomings.

Limitations and recommendations for further research

A number of limitations in our approach should be addressed. Firstly, the assumptions on which the model relies form a limitation. In particular, the training, tuning, and testing of the model depend heavily on the labels assigned to the chunks of the letters: "Paul" for the chunks of the undisputed Pauline letters and "Other" for the selected impostor letters. It is possible that these labels do not fully reflect reality, as scholarly consensus is not absolute, and the binary classification approach cannot account for possible complexities in authorship, such as the role of secretaries or co-authors. Therefore, there is a risk that the model is learning patterns based on incorrect or oversimplified assumptions, which could make its predictions unreliable.

Secondly, the limited size of Subset A should be acknowledged, i.e., the limited sizes of the training and test sets. While the full dataset was not large, the decision to keep four letters out of Subset A to create Subset B further reduced the size of the training and test sets. This decision allowed for the evaluation of the model's performance on entirely unseen letters, offering valuable insights, but it decreased the diversity and size of the training and test sets. The limited size of the test sets may have affected the reliability of the model evaluations, and the limited size of the training sets may have affected the robustness and generalization of the model variants. With the model variants having

limited examples to learn from, there was an increased risk of overfitting, signs of which we have seen in our results. Additionally, the influence of random factors, such as the initialized weights and the specific training-test split, may have been amplified due to the limited size of the training sets. Efforts were made to mitigate these challenges, including employing cross-validation and dropout regularization. However, additional strategies, such as averaging the results of multiple runs with different random seeds, could have improved the robustness and generalization of the model.

Thirdly, the study's focus on a single model is a significant limitation. It should be noted that the model developed in this study is by no means the only model that can be used for this authorship verification problem. Our conclusions should thus be interpreted as interim results and can serve as a starting point or even benchmark for future research on this topic with more sophisticated models. Choices regarding the model type, as well as its particular structure and settings, can be debated. For instance, one might question whether more powerful models, such as Transformers, are more suitable for this task, especially considering the limited size of the dataset. Transformers have shown remarkable capabilities in text classification tasks, surpassing the performance of LSTM models (Gasparetto et al. 2022). They excel in their ability to model complex patterns and capture long-range dependencies within texts, which could be beneficial in authorship verification tasks. Another suggestion for further research is to explore the potential benefits of using pre-trained word embeddings instead of a trainable embedding layer. Such pre-trained word embeddings, trained on a substantial amount of text data, capture syntactic and semantic relations between words and could potentially improve the model's performance. Furthermore, incorporating more linguistic features into the model, such as part-of-speech tags, can provide additional information for the model to learn from. This might help the model to recognize patterns specific to an author's writing style. It should be noted that the toolbox of modern AI research is developing at such a pace that the methods used in this paper are already dated. However, our approach to this particular problem, in particular our selection and preprocessing of the training data, can serve as a model for future research with the newest AI techniques. It is quite possible that such new research will lead to completely opposite results with respect to the present paper.

Lastly, we note that this study leaves an important question unanswered: What exactly is our model learning from the training data? What aspects does it consider to be indicative of Paul's writing? From the obtained results, we cannot tell why the model classified chunks as Pauline or non-Pauline, so we do not know which features or patterns the model is relying on for classification. This highlights another potential area for further research: explainable artificial intelligence (XAI) techniques. XAI aims to improve the transparency and interpretability of AI systems, offering insights into their decision-making processes and the factors influencing these decisions (Xu et al. 2019). Thus, in the context of our authorship verification problem, XAI methods could help to identify linguistic features that the model considers to be indicative of Paul's writing. For example, attention mechanisms, used in Transformers, might be useful in this context. Attention mechanisms enable models to selectively focus on crucial parts of the input data, thereby potentially improving both performance and interpretability of the model (El Houda Dehimi and Tolba 2024). By employing attention mechanisms, or other XAI methods, we may identify specific features contributing to the classification of text chunks

as Pauline or non-Pauline. This could lead to valuable insights into the authorship verification problem and could inform further model refinement.

Despite the described limitations, this study offers a deep learning approach to the authorship verification problem of the disputed Pauline letters that might serve as a model for further research. While caution is warranted in drawing strong conclusions from the obtained results, the accuracy scores of 0.84 and 0.83, combined with the model variants recognizing the vast majority of chunks from Galatians and Philemon as Pauline, and the consistency of the plaintext variant's results with current scholarship, show considerable promise. Further research based on the presented model might not only give more insight into the authorship verification problem of the disputed Pauline letters, but could also inform approaches to other text classification problems, especially those involving Ancient Greek texts.

Acknowledgements

The authors thank the editor, the two anonymous referees, and Marius Dorobantu for helpful comments on an earlier version of the article. This article is based on Evy Beijen's Bachelor's thesis in Mathematics at the Vrije Universiteit Amsterdam. The authors also thank Peter-Ben Smit for brainstorming the topic and approach of the thesis project; Sandjai Bhulai for giving feedback on the models and their implementation; and Willem van Peursen for giving advice on creating an article out of the thesis, and the advice to submit it to *HIPHIL Novum*.

Declaration of funding

Rianne de Heide's work was supported by NWO Veni grant number VI.Veni.222.018. Evy Beijen did not receive specific funding for this research.

Bibliography

Abadi, Martín, et al. 2015. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. Accessed May 8, 2024. https://www.tensorflow.org/.

Chollet, François, et al. 2015. *Keras*. Accessed May 8, 2024. https://keras.io.

Ehrman, Bart D. 2020. *The New Testament: A Historical Introduction to the Early Christian Writings*. Oxford: Oxford University Press.

El Houda Dehimi, Nour, and Zakaria Tolba. 2024. "Attention Mechanisms in Deep Learning: Towards Explainable Artificial Intelligence." In *2024 6th International Conference on Pattern Analysis and Intelligent Systems (PAIS)*, 1–7. https://doi.org/10.1109/PAIS62114.2024.10541203.

Gasparetto, Andrea, Matteo Marcuzzo, Alessandro Zangari, and Andrea Albarelli. 2022. "A Survey on Text Classification Algorithms: From Text to Predictions." *Information* 13 (2): 83. https://doi.org/10.3390/info13020083.

Graves, Alex, and Jürgen Schmidhuber. 2005. "Framewise Phoneme Classification with Bidirectional LSTM and Other Neural Network Architectures." *Neural Networks* 18 (5): 602–610. https://doi.org/10.1016/j.neunet.2005.06.042.

Hochreiter, Sepp, and Jürgen Schmidhuber. 1997. "Long Short-Term Memory." *Neural Computation* 9 (8): 1735–80. https://doi.org/10.1162/neco.1997.9.8.1735.

Johnson, Kyle P., Patrick J. Burns, John Stewart, Todd Cook, Clément Besnier, and William J. B. Mattingly. 2021. "The Classical Language Toolkit: An NLP Framework for Pre-Modern Languages." In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing: System Demonstrations*, 20–29. Association for Computational Linguistics. https://doi.org/10.18653/v1/2021.acl-demo.3.

Kingma, Diederik P., and Jimmy Ba. 2017. "Adam: A Method for Stochastic Optimization." arXiv:1412.6980.

Koppel, Moshe, and Yaron Winter. 2014. "Determining If Two Documents Are Written by the Same Author." *Journal of the Association for Information Science and Technology* 65 (1): 178–87. https://doi.org/10.1002/asi.22954.

Lake, Kirsopp, ed. 1913. *The Apostolic Fathers*. Christian Classics Ethereal Library. Accessed March 2, 2024. https://www.ccel.org/l/lake/fathers/toc.htm.

Martinez-Cantin, Ruben. 2014. "BayesOpt: A Bayesian Optimization Library for Nonlinear Optimization, Experimental Design, and Bandits." *Journal of Machine Learning Research* 15 (1): 3735–39.

Mealand, David L. 1989. "Positional Stylometry Reassessed: Testing a Seven Epistle Theory of Pauline Authorship." *New Testament Studies* 35 (2): 266–86. https://doi.org/10.1017/S0028688500024656.

Mealand, David L. 1995. "The Extent of the Pauline Corpus: A Multivariate Approach." *Journal for the Study of the New Testament* 18: 61–92. https://doi.org/10.1177/0142064X9601805904.

Metzger, Bruce M., and Bart D. Ehrman. 2005. *The Text of the New Testament: Its Transmission, Corruption, and Restoration*. Oxford: Oxford University Press.

Morton, A. Q. 1965. *The Authorship of the Pauline Epistles: A Scientific Solution*. Saskatoon: University of Saskatchewan.

Morton, A.Q., and A.D. Winspear. 1971. *It's Greek to the Computer*. Montreal: Harvest House.

Perry, Jordan. 2021. "Examining the Authenticity of Plato's Epistle VII through Deep Learning." Bachelor's thesis, Harvard University.

Roy, Ashley, and Paul Robertson. 2023. "Applying Cosine Similarity to Paul's Letters: Mathematically Modeling Formal and Stylistic Similarities." In *New Approaches to Textual and Image Analysis in Early Jewish and Christian Studies*, edited by Claire Clivaz and Ken M. Penner, 88–117. Leiden: Brill.

Savoy, Jacques. 2019. "Authorship of Pauline Epistles Revisited." *Journal of the Association for Information Science and Technology* 70 (10): 1089–97. https://doi.org/10.1002/asi.24176.

Sari, Yunita. 2018. "Neural and Non-neural Approaches to Authorship Attribution." PhD diss., University of Sheffield.

Society of Biblical Literature (SBL). 2023. *SBL Greek New Testament*. GitHub. Version 1.2. Accessed February 24, 2024. https://github.com/LogosBible/SBLGNT.

Snoek, Jasper, Hugo Larochelle, and Ryan P. Adams. 2012. "Practical Bayesian Optimization of Machine Learning Algorithms." In *Advances in Neural Information Processing Systems* 25, edited by F. Pereira, C. J. Burges, L. Bottou, and K. Q. Weinberger. Curran Associates, Inc.

Srivastava, Nitish, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. "Dropout: A Simple Way to Prevent Neural Networks from Overfitting." *Journal of Machine Learning Research* 15 (56): 1929–58.

Van der Ventel, Brandon I.S., and Richard T. Newman. 2022. "Application of the Term Frequency-Inverse Document Frequency Weighting Scheme to the Pauline Corpus." *Andrews University Seminary Studies* 59 (2): 251–71.

Xu, Feiyu, Hans Uszkoreit, Yangzhou Du, Wei Fan, Dongyan Zhao, and Jun Zhu. 2019. "Explainable AI: A Brief Survey on History, Research Areas, Approaches and Challenges." In *Natural Language Processing and Chinese Computing (NLPCC 2019),* edited by Jie Tang, Min-Yen Kan, Dongyan Zhao, Sujian Li, and Hongying Zan, 563–74. Cham: Springer. https://doi.org/10.1007/978-3-030-32236-6_51.

Yu, Yong, Xiaosheng Si, Changhua Hu, and Jianxun Zhang. 2019. "A Review of Recurrent Neural Networks: LSTM Cells and Network Architectures." In *Neural Computation*, 31 (7): 1235–70. https://doi.org/10.1162/neco_a_01199.

Appendix

Python code for this project is available at https://github.com/evybeijen/DisputedPaulineLetters.