Classifying Political Bias in German News Articles

Sona Mehdizade
Freie Universität Berlin
Berlin, Germany
sonam00@zedat.fu-berlin.de

ACM Reference Format:
Sona Mehdizade. . Classifying Political Bias in German News Articles. In . , 6 pages. https://doi.org/

Introduction

In the given project, the aim is to classify political bias (left, center, and right) in German news articles. The dataset contains articles with titles and bodies labeled as follows:

- 0: Left-leaning articles
- 1: Center articles
- 2: Right-leaning articles

As an initial step, 43 duplicate entries were identified and removed from the dataset to ensure accuracy of the analysis. The exploratory data analysis will begin by examining the balance of the target variable, followed by further insights with the statistics and vocabulary analysis.

Extracting Insights from Data

1.1 Statistical Insights

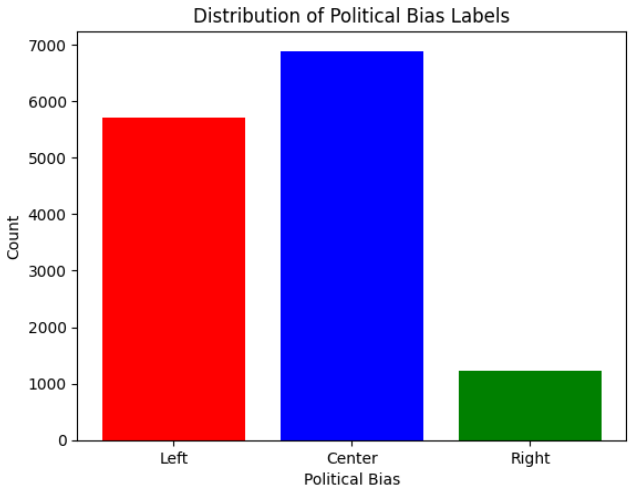

Figure 1: The distribution of the target into classes

As it is seen from Figure 1, the data is imbalanced, with the majority of the articles (6,889) categorized as center, followed by left-leaning articles (5,705), and the minority being right-leaning articles (1,218). This imbalance may affect the performance of our classification models, as they could become biased towards the majority class. Therefore, later we need to utilize techniques such as oversampling, undersampling, or SMOTE to handle this imbalance issue.

Next, we analyze the distribution of average number of words in the body of articles.

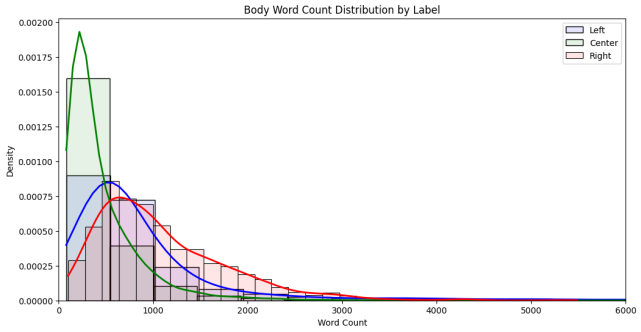

Figure 2: Body Word Count Statistics by Label

Table 1: Body Word Count Statistics by Label

Label	Mean	Std
Left	1,034.31	1,551.90
Center	465.19	586.00
Right	1,083.06	704.14

As shown in Figure 2 and Table 1, right-leaning articles have the highest average word count(1,083.06) in the body, suggesting they tend to be longer and possibly more detailed. Center-leaning articles are notably shorter with average of 465.19 words, which might indicate a more concise style of writing for this category. An insightful finding from these statistics is that the left-leaning articles exhibit a significantly high standard deviation(1,551.90) in average word count, indicating a wide variation in the length of these articles. This could imply that left-leaning articles vary greatly in their level of detail and depth of coverage, ranging from very short to very long samples.

Next, moving to the analysis of average sentence length in terms of word count whose results are demonstrated in Table 2 , we notice that right-leaning articles have the shortest titles with an average of 6.53 words in a sentence, indicating a preference for brevity in headings. Interestingly, the body text of right-leaning articles features the longest sentences(17.60 word per sentence), suggesting a tendency towards more complex and detailed writing in bodies.

Following this, we examine the average number of sentences in the body text of news articles. According to Figure 3, we notice

, ,
. ACM ISBN
https://doi.org/

Table 2: Average Sentence Lengths in Titles and Bodies by Categories

Category	Title	Body
Left	7.14	15.82
Center	7.25	15.52
Right	6.53	17.60
Total	7.14	15.82

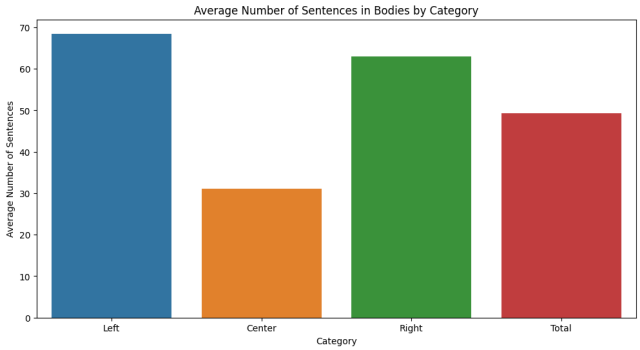

Figure 3: Average Number of Sentences in Body by Label

that left-leaning articles have the highest average sentence count at 68.48, indicating a more extensive narrative style. Right-leaning articles also exhibit a high average sentence count at 62.97, suggesting detailed and elaborate content. Center-leaning articles have a significantly lower average sentence count at 31.01, pointing to a more concise writing approach.

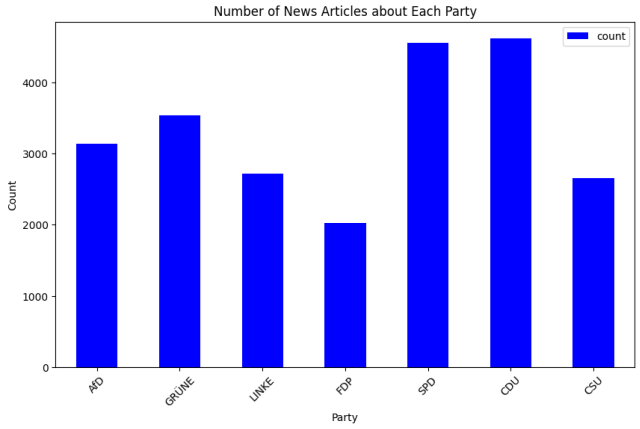

Figure 4: The number of articles mentioning main German parties

Then, we look at how many number of these articles mentioned current main political parties in Germany. We observe from Figure 4 that, CDU receives the highest attention with 4,623 articles, closely followed by SPD with 4,562 articles. GRÜNE also receive substantial coverage with 3,538 articles. In contrast, LINKE and CSU are covered less, with 2,714 and 2,655 articles respectively. AfD and FDP have moderate coverage with 3,143 and 2,029 articles respectively.

This varying levels of media focus on different political parties can be explained by the target distribution of the dataset. Center-leaning articles, which make up the largest number of samples (6,889), might contribute to the higher coverage of center-leaning parties like the CDU and SPD.

Before continuing with vocabulary-based analysis, preprocessing was applied to ensure accuracy. Analyzing raw text without preprocessing can lead to inflated unique word counts due to different forms of the same word appearing as separate entries or some with punctuation marks, skewing the results. Moreover, for n-gram analysis, preprocessing helps focus on the main words by removing stopwords, allowing us to identify key topics rather than common words that do not contribute to the understanding of the main themes. The specific preprocessing techniques used will be discussed in the next section.

1.2 Vocabulary-Based Analysis

We start with the analysis of unique word counts by categories, whose results are demonstrated in Table 3 below. It can be concluded that right-leaning articles, which also have the fewest samples in the dataset, consequently show fewer unique words, with 3,153 and 84,746 unique words in title and body, respectively. An interesting insight from these statistics is that, although the center-leaning category has 1,184 more samples in the dataset than the left-leaning category, the left-leaning category has 33,000 more unique words than the center-leaning category, which highlights the specificity and richness of the vocabulary used by the left-leaning category.

Moreover, the total number of unique words across all categories is 17,871 for titles and 240,008 for bodies, which is significantly greater than the largest unique word count for any single category. This indicates that the vocabulary used in each category is quite distinct, suggesting that each has a specific style and lexical choices.

Table 3: The number of unique words in titles and bodies by categories

Category	Title	Body
Left	9,387	149,048
Center	11,079	116,846
Right	3,153	84,746
Overall	17,871	240,008

Next, the most frequent phrases were identified using n-gram analysis. The analysis of the most frequent tri-grams, as shown in Figure 5 below, reveals that many of these phrases are related to political figures and parties. For instance, "bundeskanzlerin angela merkel", "angela merkel cdu", “recep tayyip erdogan”, ““präsident donald trump”, “bundesinnenminister horst seehofer”, “premierministerin theresa may” are among the top tri-grams, indicating a strong media focus on prominent political personalities and parties.

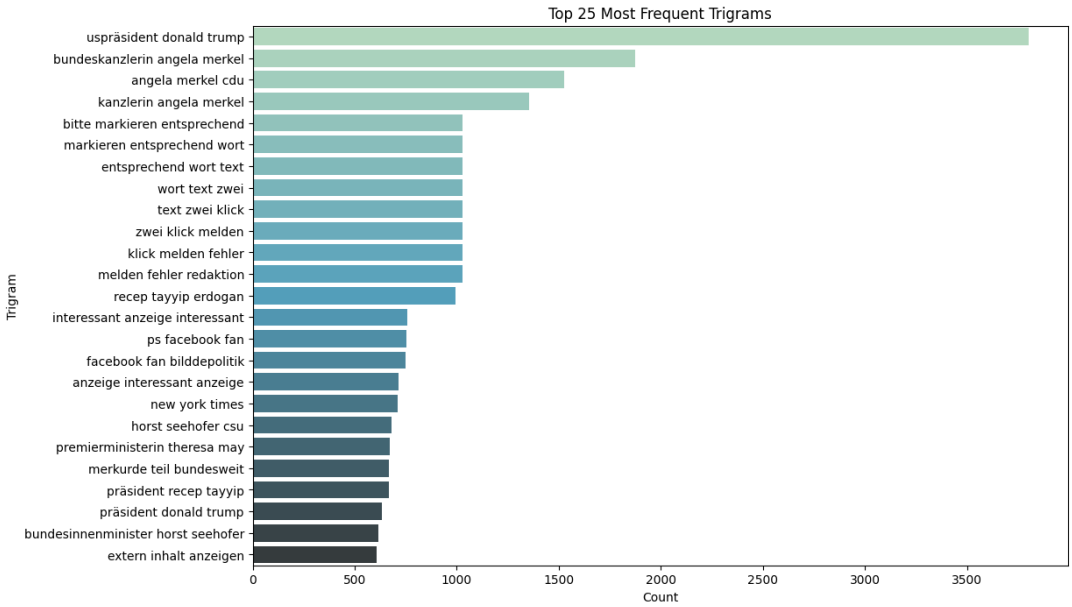

Figure 5: The most frequent trigrams in articles

Pre-processing

The chosen pre-processing steps were selected based on their relevance for German language and effectiveness in improving data quality.

(1) **Removal of Non-ASCII Characters Except German-Specific Letters**: Since dataset is in German, it includes specific characters such as 'ä', 'ö', 'ü', and 'ß'. Therefore, non-ASCII characters were removed except for German-specific letters.
(2) **Removal of Single Letter Characters**: There are no single-letter words in German. Therefore, single letter characters are typically noise in data and need to be removed.
(3) **Whitespace Normalization**: All whitespace characters were converted to single spaces to ensure consistency.
(4) **Punctuation Removal**: Punctuation removal simplifies text data and reduces complexity. Punctuation can inflate the number of tokens (e.g., "gewählt" vs. "gewählt."). This might skew word frequency counts or embeddings.
(5) **Lemmatization**: Lemmatization was preferred over stemming as it retains the original meaning of words by reducing them to their base forms using a predefined dictionary. This is particularly useful for the German language, where stemming might truncate words improperly.
(6) **Stopword Removal**: Stopwords are removed to focus on the main words that contribute to the content's meaning.

Case folding (lowercasing) was intentionally omitted as proper nouns are crucial in bias detection and some words, for example like "Morgen" (morning) vs. "morgen" (tomorrow) change meaning based on their capitalization.

Text Classification

In this task, various preprocessing and vectorization methods were applied. The preprocessing steps applied to all text data included:

(1) Removal of single letter characters.
(2) Conversion of all whitespace characters to single spaces.
(3) Removal of punctuation marks (including special characters).

Additionally, the following optional preprocessing combinations were tested:

(1) Basic preprocessing (as described above).
(2) Basic preprocessing plus removal of stopwords and non-ASCII characters.
(3) Basic preprocessing plus lemmatization, removal of stopwords, and non-ASCII characters.

To address the imbalance in the target classes, undersampling was applied to balance the training data.

The vectorization methods tested included:

(1) **Count Vectorizer** transforms text into a matrix of token counts. Each document is represented as a vector with the count of each word from the vocabulary appearing in the document.
(2) **TF-IDF Vectorizer** converts text to a matrix of TF-IDF features, which reflect both the importance of a word in a specific document and its rarity across the corpus.
(3) **N-gram Vectorizer** is similar to the Count Vectorizer but includes n-grams (contiguous sequences of n words), capturing more context than single words.
(4) **Word2Vec** generates word embeddings based on the context in the corpus, producing dense vector representations of words. Normalization was applied to these vectors for the Naive Bayes model to handle non-negative values.

Moreover, the following two models were trained and evaluated:

(1) **Naive Bayes Model:** A probabilistic classifier based on Bayes' theorem. It assumes independence between features, which simplifies the computation and is particularly effective for text classification tasks.
(2) **Feed-Forward Neural Network Model:** A multi-layer perceptron(MLP) classifier with specific hyperparameters (hidden layer size 50, batch size 64 with early stopping) was used.

The results of the models with different vectorization and preprocessing combinations are summarized in Table 4 on the next page.

From the results presented in Table 4, several key observations can be made regarding the performance of different vectorization methods and preprocessing combinations:

Firstly, The TF-IDF and N-gram vectorizers generally performed better than the Count Vectorizer (Bag of Word) and Word2Vec across both models. The N-gram vectorizer showed the highest F1 score for both Naive Bayes (0.753) and Neural Network (0.819). Word2Vec did not perform as well as the other methods in this specific task, possibly due to the need for extensive fine-tuning and large amounts of data to capture meaningful embeddings.

With regards to preprocessing combination, removing stopwords and non-ASCII characters generally improved performance slightly for most vectorization methods. On the other hand, lemmatization, combined with stopwords and non-ASCII character removal, did not significantly improve the performance for most models, and in some cases, slightly decreased it. This suggests that while lemmatization helps in reducing the dimensionality of the text, it might also remove some contextual nuances important for classification.

Considering model choices, the Feed-Forward Neural Network(FFNN) model generally outperformed the Naive Bayes model across all preprocessing and vectorization combinations. The best performance for FFNN model was with the N-gram vectorizer (F1 score of 0.819). The best performance for the Naive Bayes model was also with the N-gram vectorizer (F1 score of 0.753). This overall trend may indicate that more complex models, like the neural network based ones, benefit significantly from richer vector representations like those provided by the N-gram vectorizer.

Error Analysis

Table 5: Error Statistics for Naive Bayes

Category	TP	FP	FN	Error Rate
Left	185	44	79	16.83%
Center	167	52	69	16.55%
Right	213	70	18	12.04%

For the analysis of errors, we focus on models trained with the TF-IDF vectorization method and all preprocessing techniques applied. Examining the error statistics presented in Table 5, we observe that the The Naive Bayes model shows a higher error rate for the "Left" and "Center" categories, 16.83% and 16.55%, respectively. Considering error statistics for FFNN model in Table 6, the model

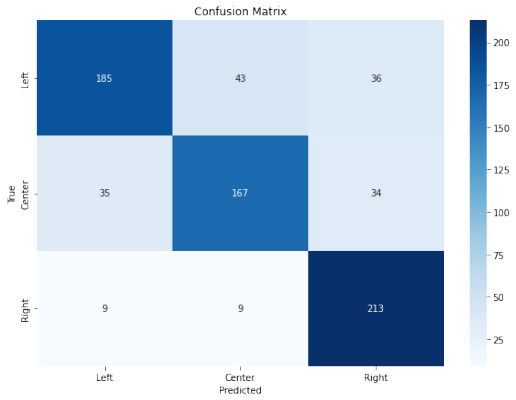

Figure 6: Confusion matrix for Naïve Bayes model

demonstrates a lower error rate across all categories compared to the Naive Bayes model.

Table 6: Error Statistics for MLP Classifier

Category	TP	FP	FN	Error Rate
Left	219	40	45	11.63%
Center	194	31	42	9.99%
Right	213	34	18	7.11%

Interestingly, from the confusion matrix of both models shown in Figure 6 and 7, the false positives and false negatives between these two categories indicates that both models struggled with distinguishing between left and center articles, which may share similar vocabulary and topics. Therefore, this confusion may be attributed to the overlapping themes and vocabulary used in articles from these categories.

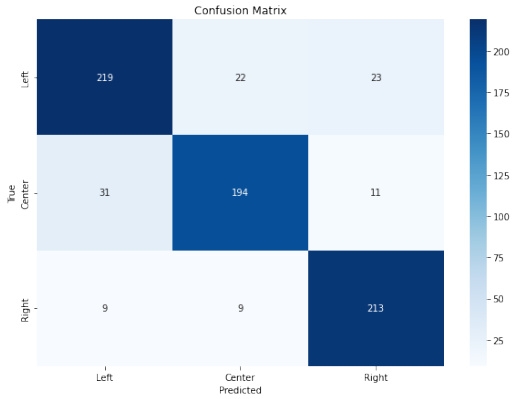

Figure 7: Confusion matrix for MLP (NN) model

The examination of several samples of falsely predicted articles for each model reveals that they are often outliers in terms of text length. Articles with significantly higher or lower word counts than

Table 4: Model Performance with Different Vectorization and Preprocessing Combinations

Vectorization Method	Preprocessing	Naive Bayes F1	Neural Network F1
Count Vectorizer	Basic preprocessing	0.727	0.808
Count Vectorizer	Stopwords & ASCII removal	0.726	0.810
Count Vectorizer	Lemmatization, Stopwords & ASCII removal	0.719	0.808
TF-IDF Vectorizer	Basic preprocessing	0.732	0.818
TF-IDF Vectorizer	Stopwords & ASCII removal	0.740	0.808
TF-IDF Vectorizer	Lemmatization, Stopwords & ASCII removal	0.734	0.814
N-gram Vectorizer	Basic preprocessing	0.753	0.819
N-gram Vectorizer	Stopwords & ASCII removal	0.754	0.815
N-gram Vectorizer	Lemmatization, Stopwords & ASCII removal	0.747	0.810
Word2Vec	Basic preprocessing	0.511	0.661
Word2Vec	Stopwords & ASCII removal	0.544	0.708
Word2Vec	Lemmatization, Stopwords & ASCII removal	0.549	0.715

the average for their category are more likely to be misclassified. In the exploratory analysis, we discussed that while the left-leaning category has a high average word count, it also has a large standard deviation. This variability might confuse the model, leading to the misclassification of shorter-than-usual center-leaning articles as right-leaning, and longer-than-usual center-leaning articles as left-leaning.

Moreover, the potential grouping of political biases (according to German political parties' leaning) such as "Far-left," "Center-left," "Center," "Center-right," and "Far-right" can blur distinctions and make it challenging for models to differentiate between categories. Articles that fall on the border between these biases may share significant linguistic and thematic similarities, contributing to misclassification.

Textual Similarity

In this task, a distributional semantics approach was used to measure the semantic textual similarity between 15 randomly selected sentences from the dataset. The steps involved were as follows: first, 15 sentences were randomly selected from the dataset. Then, these sentences were cleaned without lemmatizing to ensure the text was free of noise, allowing the similarity measure to focus on the meaningful content of the sentences. Next, a Word2Vec model was trained on the cleaned sentences, and the average word vectors for each sentence were computed. Finally, cosine similarity was calculated between each pair of sentences to determine their semantic similarity. The cosine similarity matrix for the 15 sentences is shown in Figure 8.

The highest correlation is between Sentence 2 and Sentence 11, with a score of 0.33. The respective sentences in uncleaned form are:

> **Sentence 2:**
> Der Bundestag verabschiedete am Donnerstagabend zwei Gesetze, die unter anderem steuerliche Übergangsregelungen vorsehen und für Rechtssicherheit an den Finanzmärkten sorgen sollen - auch im Fall eines harten Brexits.
>
> **Sentence 11:**
> Brexit: EU-Kommission plant für „schlimmstes Szenario“ 15.41 Uhr: Die EU sieht sich gut für den Fall eines ungeregelten Brexits gewappnet.

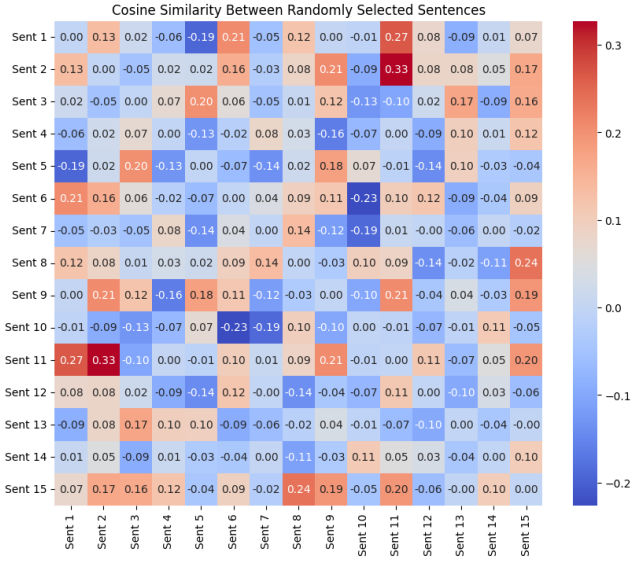

Figure 8: Cosine similarity matrix for 15 selected sentences

As it is clear from the sentences, this high similarity can be attributed to the common topic of Brexit, discussed in both sentences.

Textual Similarity - Bonus Task

For the final task, two pre-trained BERT models were fine-tuned: one for binary classification and one for multiclass classification. Given that the "center" samples are the majority class, they were taken as a separate category for binary classification, while "left" and "right" samples were grouped together. Therefore, the binary classification model distinguishes between "center" and "left/right" articles, while the multiclass classification model categorizes articles into "left," "center," or "right."

The German BERT base model from the Hugging Face Transformers library ('dbmdz/bert-base-german-uncased') was used. The

training process involved tokenizing the text data and splitting it into train and test sets.

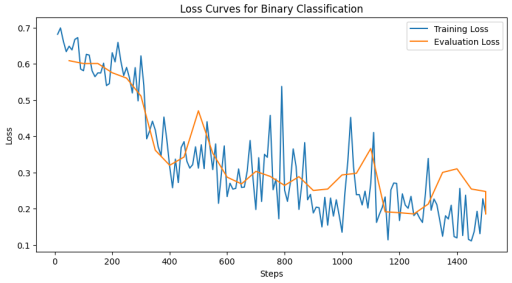

Figure 9: Loss Curve for Binary Classifier

To prevent overwriting the current weights too much, hyperparameters were carefully selected, such as a smaller learning rate and fewer epochs, to ensure that fine-tuning was subtle. The hyperparameters included five epochs, a batch size of 16, a default learning rate with a weight decay of 0.01, 500 warmup steps, and mixed precision training (fp16). Weight decay was used to regularize the model by discouraging excessively large weights, thereby helping to prevent overfitting. Early stopping was also implemented with a patience of five to halt training once no further improvement was observed, thus avoiding overfitting.

The binary classification model outperformed the multiclass model with an F1 score of 0.91 compared to 0.84. This difference can be attributed to the complexity of distinguishing between three classes in the multiclass setup versus the simpler binary classification. Additionally, grouping "left" and "right" samples together for binary classification likely reduced misclassification errors that could arise from the nuances between these categories.

Since in this case undersampling is applied to handle target balance, the number of samples in each class might not be sufficient for multiclass model, thus more data for tuning might be needed to achieve higher performance.

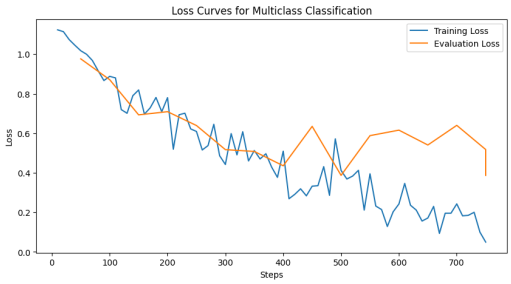

Figure 10: Loss Curve for Multiclass Classifier

The loss curves for both models indicated a generally good fit, with the training loss steadily decreasing and the evaluation loss showing some fluctuations.