Experiment with halucination detection for LLM Q-A pairs

Jurgen de Vries (Studentnr: 852759710)

January 2026

1 Introduction

1.1 Detecting Hallucination in LLM's

Large language models (LLMs) are prone to generating hallucinated responses that appear fluent but contain incorrect or unsupported information. Detecting such hallucinations is an important challenge, particularly in applications where reliability and trust are critical. In addition to achieving good classification performance, it is essential to understand why a model predicts a response as hallucinated.

Text classification tasks have traditionally relied on interpretable models such as TF-IDF (Term Frequency Inverse Document Frequency) combined with LR (Logistic Regression), where feature weights provide direct insight into which words influence predictions. More recently, transformer-based models such as BERT (Bidirectional Encoder Representations from Transformers) have achieved strong performance by leveraging contextual representations, but their decision processes are less transparent and typically require post-hoc explanation methods.

In this experimental report, I compare a LR classifier and a BERT-based classifier for hallucination detection. LR is used as an interpretable baseline, with model coefficients analyzed to identify the most influential lexical features. BERT is evaluated in terms of classification performance and explained using LIME (Local Interpretable Model-agnostic Explanations) to obtain word-level explanations for individual predictions.

This is related to the course because I compare different models and use the underlying workings of LR as a way to explain why the model is taking a decision. Furthermore I use different techniques to not only clean the data to remove less useful features but also try to discover patterns in the data using various techniques as dimensionality reduction and outlier detection.

1.2 Related work

LR came out as the best performing model compared to KNN, Passive Aggressive classifier and Naïve Bayes Classifier in the study of Adeyiga et al [1] predicting fake news. They used text cleaning to get consistency in used words and remove meaningless words. After that TF-IDF was used to penalize words that are common in all documents, they achieved scores above 96% for accuracy. Therefore I will use LR in combination with TF-IDF as a baseline model and see if this could work on detecting hallucination of LLM's as well.

Recent work presents a framework for interpreting BERT's predictions using both LIME and SHAP (SHapley Additive exPlanations), comparing these explainers in terms of local fidelity, global consistency and computational cost, and demonstrating that model-agnostic methods can produce intuitive case-specific explanations for BERT classifiers [2]. I will try to fit a BERT model to predict hallucination on the dataset and make it explainable using LIME.

2 Goal

The goal of this experiment is to compare and explore how different modeling approaches identify hallucinated content. By examining a transparent linear model alongside a high-capacity neural model supplemented with post-hoc explanations, this study aims to uncover patterns in the data, gain insight into model behavior, and support exploratory data analysis of hallucination-related linguistic features.

3 Data analysis

3.1 Dataset

The used dataset can be found at `https://www.kaggle.com/competitions/ml-olympiad-detect-hallucinations-in-llms/data`. I used the train.csv dataset as this is the dataset that contains a target feature. The dataset consists of 16668 records. This will be sufficient to split into train and test sets. Few records of hallucinations occur in the dataset 894 versus 15793 for not hallucinating. Predicting "not hallucinating" for all records would therefore be around 94.6% accuracy, but this is not a valid metric for a dataset as unbalanced as this. Looking at recall and and precision, all hallucinations would be predicted as no hallucination, scoring 0 and resulting in an F1 score of 0 (see equations 1, 2 and 3). The aim is to get better scoring models for this imbalanced dataset. The dataset contains 4 columns namely Id, Prompt, Answer, Target. Id is not useful in this case, Prompt and Answer should be used to predict the target column.

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}, \tag{1}$$

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{2}$$

$$\text{F1} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \tag{3}$$

3.2 Pre-processing

With two text columns there are no comparable features. Also, a lot of words might not be useful for classification. The first task is to clean the dataset. Words like "Hello" and "hello" are the same, so to reduce the unique number of words the first step is to transform all text to lowercase. To further reduce the number of unique words in the data, words containing numerical characters are removed, as such words often correspond to non-semantic identifiers and are not informative for hallucination detection. This cuts the number of unique words from 188015 to 64981. Rows that become empty after cleaning for Prompt or Answer are removed, after this process, 16195 rows remain of which 807 hallucinations.

Training an LR model on just text fragments would be impossible. Therefore I converted the text to TF-IDF. This is applied separately to the Prompt and Answer columns and than the resulting arrays are combined, forming the feature set for LR. The TF-IDF (see equation 6) is applied separately to Prompt and Answer to keep these features distinguishable, otherwise a word that is found in both prompt and answer would be mapped to the same feature. Words occurring less than 10 times and words occurring in 90% or more of the documents are filtered out here as well as english stop words. This results in 14514 features.

AS a second approach I tried to further reduce the unique words by applying tokenization and stemming on the raw dataset. This results in a second dataset with 10531 features.

$$\text{TF}(t, d) = \frac{f_{t,d}}{\sum_{t' \in d} f_{t',d}} \tag{4}$$

$$\text{IDF}(t, D) = \log \frac{N}{1 + |\{d \in D : t \in d\}|} \tag{5}$$

$$\text{TF-IDF}(t, d, D) = \text{TF}(t, d) \cdot \text{IDF}(t, D) \tag{6}$$

3.3 Data analysis

I used LDA (Linear Discriminant Analysis) for dimensionality reduction to view separation in the classes. Since this is not suitable for large or sparse feature spaces I first applied Truncated SVD (Singular Value Decomposition, SVD finds the singular values that capture the most important information in the data, reducing dimensionality while preserving the main patterns. The KDE-plot of the classes after LDA shows how the features (or LDA components) of both classes

overlap, but also shows separation of feature densities between the classes, this was tested for different values of n_components. So it looks like the hallucination has some indicative features (see figure 1). It also show that increasing the number over components in SVD leads to improved class separation in LDA projection. The further tokenized and stemmed dataset shows the same results and will be selected for further exploration as this holds less features, but what looks like the same information (see figure 2).

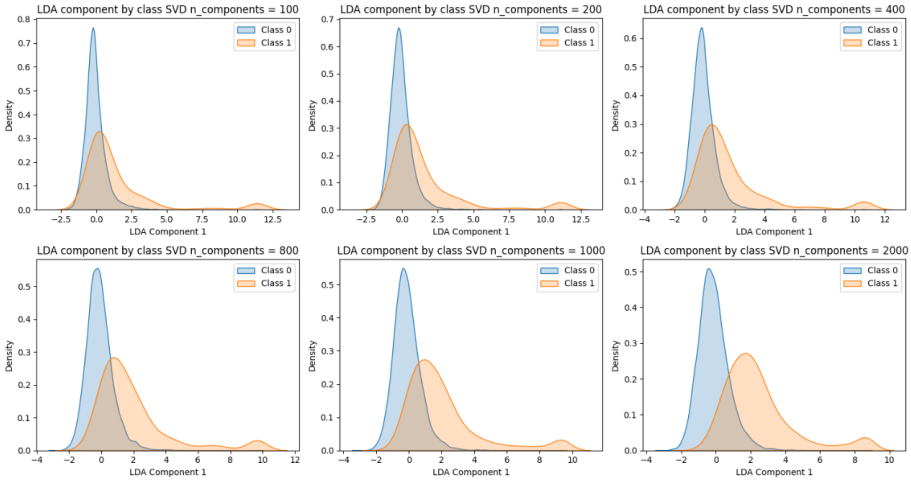

Figure 1: kde-plots after applying Truncated SVD and LDA on the cleaned dataset

Training LDA as an experimental classifier shows high accuracy (94%) as it probably mostly predicts "no hallucination", but precision, recall and F1 tend to stay low, 45% and 19% respectively. However it slowly increases using more components. Also the roc_auc shows that the model is learning a bit (see table 1. More components might improve the results a bit. But this model makes it hard to explain the decisions, so I will not further explore this.

SVD components	Accuracy	Precision	Recall	F1-score	ROC AUC
100	0.9488	0.4510	0.1429	0.2170	0.7079
200	0.9478	0.4310	0.1553	0.2283	0.7191
400	0.9491	0.4688	0.1863	0.2667	0.7217
800	0.9438	0.3735	0.1925	0.2541	0.7185
1000	0.9472	0.4342	0.2050	0.2785	0.7168
2000	0.9432	0.3708	0.2050	0.2640	0.7129

Table 1: Performance of a Linear Discriminant Analysis classifier after dimensionality reduction with Truncated SVD for varying numbers of retained components. All values are rounded to four decimal places.

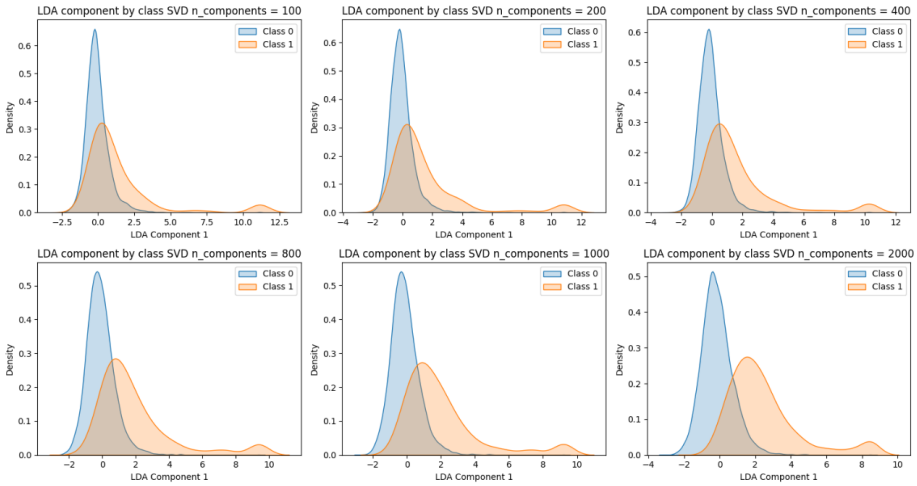

Figure 2: kde-plots after applying Truncated SVD and LDA on the stemmed and tokenized dataset

4 Outlier detection

I used LOF (Local Outlier Factor) on the TruncatedSVD features (50 components) to detect outliers in the dataset. It finds 810 outliers because of the 5% contamination setting used. Plotting inliers, outliers and the true hallucinations by reducing to two dimension with t-SNE, there are some areas where no outliers and hallucinations are found, but no clear separation or insight appeared. This is likely because the prompts and answers are highly diverse, resulting in very sparse feature representations, so traditional outlier detection does not provide additional insight(see figure 3).

A short experiment with LOF as classifier shows it is not suitable for this problem (see table 2).

SVD Components	Accuracy	Precision	Recall	F1-score
100	0.9044	0.0420	0.0421	0.0421
200	0.9046	0.0444	0.0446	0.0445
400	0.9040	0.0383	0.0384	0.0383
800	0.9031	0.0296	0.0297	0.0297
1000	0.9027	0.0259	0.0260	0.0260
2000	0.9025	0.0235	0.0235	0.0235

Table 2: Performance of Local Outlier Factor (LOF) for hallucination detection using different numbers of TruncatedSVD components.

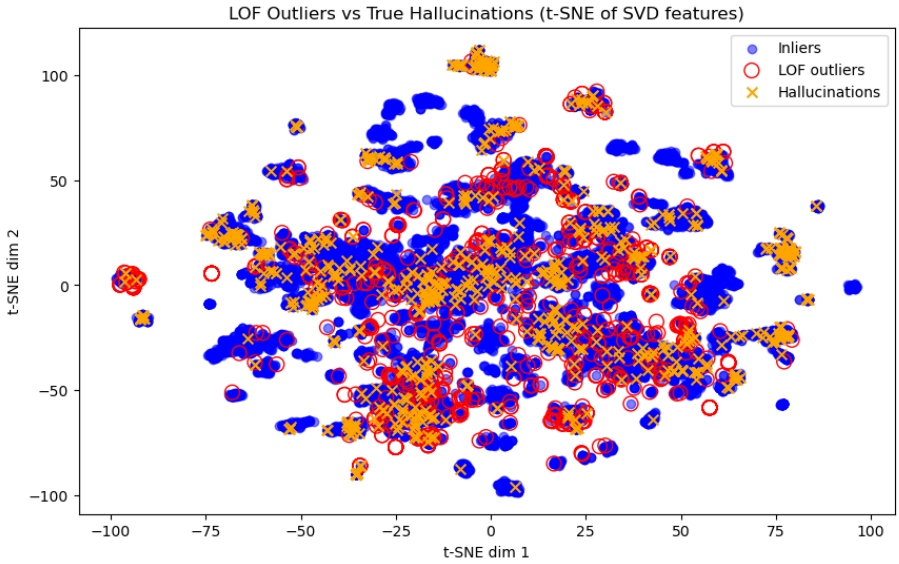

Figure 3: Scatterplot of inliers, outliers and true hallucinations after LOF.

5 Methodology and Implementation

5.1 Logistic Regression

5.1.1 Training

LR training was done on an 11th generation Intel core I7 CPU 16GB RAM (no GPU) laptop. For testing LR as a classifier I first did a quick example without any hyper parameter tuning, only setting class_weight to 'balanced' because this dataset is very imbalanced. Results are not very promising (see table 3). I do not use dimensionality reduction for training my LR model as this would become much harder to explain the outcomes. Looking at recall and the confusion matrix this model does seem to recognize at least some hallucinations.

Metric	Value
Accuracy	0.8923
Precision	0.2006
Recall	0.3913
F1 Score	0.2653
ROC AUC	0.7614

Table 3: Performance of the Logistic Regression model for a first setup on the test set.

After this I used a GridSearchCV with StratifiedKFold using 5 splits to

find the best hyper parameters for this model. Using the values in table 4. I did not use L1 regularization and saga solver due to their substantially higher computational cost in the high-dimensional feature space. For the value of C a higher value reduces regularization, allowing the model to fit sparse or rare features more freely, which can improve performance on minority classes in imbalanced datasets, further the class_weight parameters were tested. For TF-IDF I experimented with the max_features value. Therefore, I assembled a pipeline that tests the different configurations for TF-IDF and LR.

Table 4: Hyperparameter configuration for Logistic Regression with TF-IDF features

Hyperparameter	**Values**
Regularization strength (C)	{0.01, 0.1, 1, 10}
Class weighting	`'balanced', None`
Max features	{1000, 5000, 10000}
N-Gram Range	{(1,1), (2,2), (3,3)}

5.1.2 Explainability

To interpret individual predictions, we decompose the LR decision function into per-feature contributions by multiplying TF–IDF values with their corresponding learned coefficients. Features are traced back to their originating text field (prompt or answer), enabling word-level explanations of model decisions, where positive contributions increase the likelihood of hallucination, while negative contributions decrease it.

5.2 BERT

5.2.1 Training

BERT (Bidirectional Encoder Representations from Transformers) creates context-aware embeddings by attending to the entire sentence in both directions, allowing it to capture the meaning of words based on surrounding context. For tasks with a question and answer, the two texts are combined with a [SEP] token, signaling BERT to treat them as two segments so it can model their relationship while keeping them distinct, improving performance on classification or QA tasks. For training BERT models I switched to Google Colab using the T4 and A100 GPU to keep training managable. I used DistilBERT to keep model sizes and training times manageable and use a compatible DistilBERT tokenizer to tokenize the raw texts. DistilBERT has a max token length of 512 tokens so this will cut of longer texts where we could lose context. I used the base model and tokenizer called 'distilbert-base-uncased-finetuned-sst-2-english'. To address the class imbalance in the dataset, I used a weighted training approach. The WeightedRandomSampler ensures that minority-class

examples are sampled more frequently during training, while the CrossEntropy-Loss incorporates class weights to penalize misclassifications of underrepresented classes more heavily. This combination helps the model learn a more balanced decision boundary and reduces bias toward majority classes. The Dataset is split in training, evaluation and test sets in proportions 80%, 20% and 10% respectivily. For hyper paramater tuning the training and evaluation sets are split in half to keep this proces managable. For hyper parameter tuning 10 trials were performed due to time and computational costs. Hyper parameters were optimized using Optuna via the HuggingFace Trainer's built-in hyper parameter search, maximizing validation F1 score. Each trial reinitialized the model to ensure fair comparison across configurations. I ran 10 trials to find the best performing parameters, looking at learning rate, batch size, weight decay and number of epochs.

5.2.2 Explainability

To interpret predictions of hallucination generated by the BERT-based classifier, we apply LIME. For each instance, LIME constructs a local surrogate model by perturbing the input text and observing changes in the predicted hallucination probability. The resulting token-level importance scores highlight which words or phrases in the model's response most strongly contribute to a hallucination or non-hallucination decision. This enables qualitative analysis of whether the model's predictions are driven by semantically implausible, unsupported, or uncertain language, providing insight into the linguistic cues associated with hallucination behavior.

6 Evaluation and results

6.1 Logistic Regression

The best performing model after hyper parameter tuning was a model using C=1, max_features=10.000, class_weight=balanced and ngram_range (1,1). I would have expected using bigrams or trigrams would add more meaningful features, but this did not show in performance of the model. Repeating the best scoring parameters 20 times the averages don't show LR did seem able to classify the data well, assumably because of the high dimensionality and sparsity in the feature space (see table 5).

Table 5: Mean classification performance over 20 random stratified train–test splits

Metric	**Mean ± Std**
Accuracy	0.8884 ± 0.0047
Precision	0.1863 ± 0.0125
Recall	0.3693 ± 0.0296
F1-score	0.2475 ± 0.0165

Looking at the ROC-curve (4) it is better than just predicting non-hallucination always, but looking at the confusion matrix (5) it is not very promising.

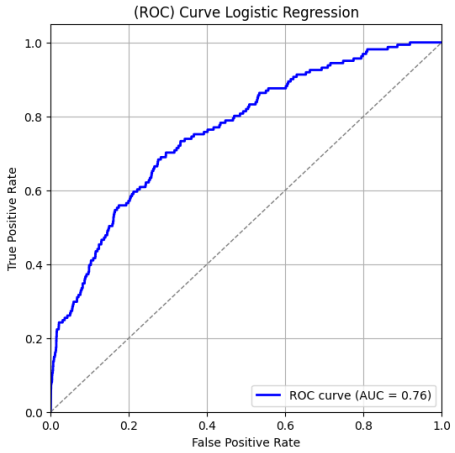

Figure 4: ROC curve Logistic Regression

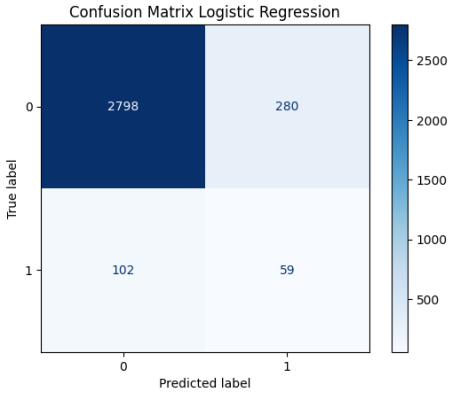

Figure 5: Confusion matrix Logistic Regression, 0 = 'No hallucination', 1 = 'Hallucination'

The results for explainability show the insights acquired by returning word level explanations, where positive sign show indication for hallucination and negative signs show indication for non hallucination. However, the model is not performing well this does show the key contributors of why the model decides for hallucination or not (see 6 for an example).

Table 6: Top contributing TF-IDF features for a sample prediction

Feature	TF-IDF	Coefficient	Contribution
PROMPT::answer	0.304	−2.023	−0.616
ANSWER::locat	0.364	−1.105	−0.402
ANSWER::therefor	0.144	−2.651	−0.383
PROMPT::reason	0.199	+1.854	+0.368
PROMPT::step	0.333	−1.067	−0.355
PROMPT::justifi	0.245	+1.185	+0.290
ANSWER::hous	0.238	−1.154	−0.274
PROMPT::ye	0.305	−0.810	−0.247
PROMPT::travel	0.239	−0.806	−0.193
ANSWER::white	0.249	−0.706	−0.176

6.2 BERT

The BERT model shows a little improvement on precision, recall and F1 scores, this is not directly visible in the confusion matrix (different test split) and ROC curve (see figures 6, 7). This becomes more visible looking at the average performance scores in table 7.

Metric	Mean ± Std
Accuracy	0.9037 ± 0.0090
Precision	0.2199 ± 0.0147
Recall	0.3205 ± 0.0484
F1-score	0.259 ± 0.0198

Table 7: Performance metrics of the hallucination detection model in BERT. Values are reported as mean ± standard deviation across multiple runs.

The results for explainability show the insights acquired by returning word level explanations, although the model is not performing well this does show the key contributors of why the model decides for hallucination or not just like LR combined with TF-IDF (see 8 for an example). LIME does show more information because we are not looking at the tokenized words but at the original text and also get an overview of where the words appear in text.

7 Conclusions and discussion

I was able to obtain results using both LR and BERT-based models, and the initial data analysis appeared promising. However, due to the imbalanced dataset and the highly sparse and high-dimensional feature space, the overall predictive performance was limited. Though BERT does performs a little better overall looking at roc_auc and mean F1 scores.

LR provides a simple and interpretable framework and trains efficiently without significant computational overhead. Nevertheless, the results were not prac-

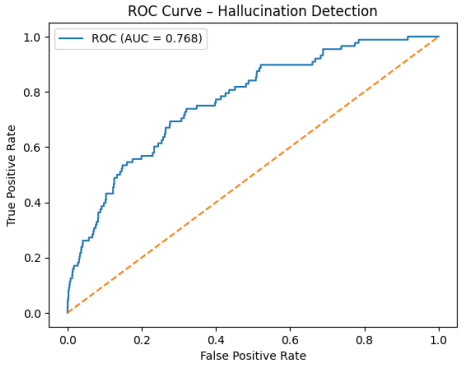

Figure 6: ROC Curve BERT

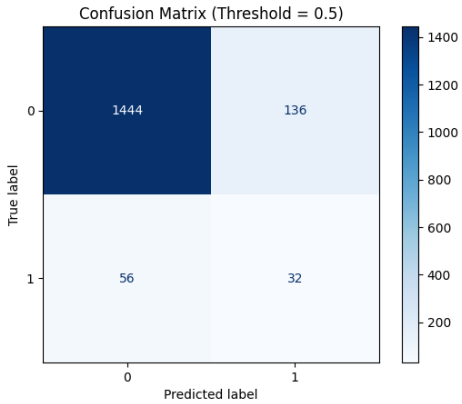

Figure 7: Confusion matrix BERT, 0 = 'No hallucination', 1 = 'Hallucination'

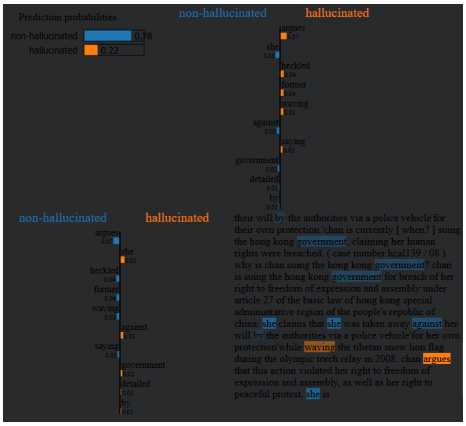

Figure 8: LIME explanations on BERT prediction

tically useful for hallucination detection. Future work could explore incorporating richer linguistic information, such as part-of-speech features or Word2Vec embeddings, to improve predictive power.

BERT offers a more powerful and flexible approach but requires substantially more computational resources, with training times ranging from minutes to hours depending on hardware. Despite experimenting with different configurations, satisfactory performance was not achieved. Additionally, BERT has a maximum sequence length of 512 tokens, and further investigation is needed to determine optimal text truncation strategies; as noted by Sun et al. [3], truncating the middle portion of the text can be particularly effective.

Addressing the class imbalance remains critical. Collecting more hallucination examples, applying bootstrapping techniques, or constructing a more evenly distributed test set could improve model performance. With additional computational resources and time, more extensive hyperparameter tuning could be performed, potentially yielding better results.

Other model architectures, such as Support Vector Machines (SVMs) or Multi-Layer Perceptrons (MLPs), could also be explored for this task. However, these models are typically less interpretable than LR, so combining them with explainability methods like LIME would be necessary to understand which features influence the predictions, particularly when the goal is to detect hallucinations reliably.

Overall, the current models are not yet suitable for deployment in real-world applications. Future work should focus on improving data balance, feature representation, and model optimization to enhance hallucination prediction performance.

8 Code

The code for both LR + TF-IDF and BERT can be found on `https://github.com/jurgendevries/llm-hallucination-detection-lr-tfidf-bert`.

References

[1] Johnson Adeleke Adeyiga, Philip Gbounmi Toriola, Temitope Elizabeth Abioye, Adebisi Esther Oluwatosin, et al. Fake news detection using a logistic regression model and natural language processing techniques. *Research Square (preprint)*, 2023.

[2] Manish Shukla. Interpreting bert using lime and shap. *Available at SSRN 5881142*, 2025.

[3] Chi Sun, Xipeng Qiu, Yige Xu, and Xuanjing Huang. How to fine-tune bert for text classification? In *China national conference on Chinese computational linguistics*, pages 194–206. Springer, 2019.