UNIVERSITÉ CÔTE D'AZUR

Détection de Fake News par Apprentissage Automatique : Approche Basée sur le Traitement Automatique du Langage Naturel

Bouslama Khalil
Loghmari Iyed

April 19, 2025

Sommaire

1. Introduction

1.1 Project Context

Fake news, or false information, refers to deliberately misleading or deceptive content often designed to manipulate public opinion, generate *buzz*, or influence major events such as political elections or health crises. Their rapid spread can have significant social, political, economic, and public health consequences. For example, during the COVID-19 pandemic, misinformation contributed to vaccine hesitancy and the circulation of ineffective or even dangerous treatments.

In this context, combating misinformation has become a critical challenge for governments, journalists, researchers, and citizens. Given the sheer volume of content, manual fact-checking is no longer sufficient. This is where artificial intelligence (AI) comes into play, particularly techniques in **Natural Language Processing (NLP)** and **Deep Learning**, which offer promising solutions to automatically detect and classify dubious content.

1.2 Project Objective

The project has two main objectives:

- Design a complete text processing pipeline, from linguistic preprocessing to semantic feature extraction using word *embeddings*;
- Build and train a *Deep Learning* model—specifically a **Long Short-Term Memory (LSTM)** network—to perform binary classification: *fake* vs. *real*.

2. Task Description

2.1 Main Task: Binary Classification (Fake vs Real)

The main task of this project is to classify news articles into two distinct categories: **Fake News** (false information) and **Real** (verified or reliable information). This is a **binary classification** task, a type of task commonly encountered in text processing and data analysis systems. The goal is to predict whether a given article is authentic or not, based solely on the textual content of the article, without considering external factors such as the source or the reputation of the author.

2.2 Data Type: News Articles Text (Title + Content)

The data used in this project comes from news articles sourced from various outlets, publicly available on Kaggle[1]. Each instance in the dataset includes both the **title** and the **full content** of the article, represented as raw text. Typically, the content consists of several paragraphs discussing a specific topic.

Challenges: Short texts, ambiguity, varying vocabulary, sometimes very similar despite opposing classes In summary, although the binary classification task may seem simple in theory, it is complicated by several factors related to the data itself, requiring advanced text analysis and modeling techniques.

[1]`https://www.kaggle.com/datasets/bhavikjikadara/fake-news-detection`

3. Jeu de données

3.1 Description

The dataset used for this project consists of two CSV files: one for **Fake News** and another for **Real News**. These files were merged during the data preprocessing step, and a new column **label** was added to indicate the class of each article. The *Fake News* articles were labeled with 0, while the *Real News* articles were labeled with 1.

The dataset includes the following columns:

- **title**: The title of the news article.
- **text**: The full textual content of the article.
- **subject**: The subject or category of the article (e.g., politics, health, etc.).
- **date**: The publication date of the article.
- **label**: The label associated with the article, where 0 corresponds to *Fake News* and 1 to *Real News*.

Titre	**Texte (extrait)**	**Label**
Donald Trump Sends Out Embarrassing New Yearâ€™s Eve Message,	This is Disturbing,"Donald Trump just couldn t wish all Americans a Happy New Year and leave it at that. Instead, he had to give a shout out to his enemies, haters and the very dishonest fake news media. The former reality show star had just one job to do and he couldn t do it. As our Country rapidly grows stronger and smarter, I want to wish all of my friends, supporters, enemies, haters, and even the very dishonest Fake News Media, a Happy and Healthy New Year, President Angry Pants tweeted...	Fake
As U.S. budget fight looms, Republicans flip their fiscal script,"WASHINGTON (Reuters)	The head of a conservative Republican faction in the U.S. Congress, who voted this month for a huge expansion of the national debt to pay for tax cuts, called himself a â€œfiscal conservativeâ€ on Sunday and urged budget restraint in 2018. In keeping with a sharp pivot under way among Republicans, U.S. Representative Mark Meadows, speaking on CBSâ€™ â€œFace the Nation drew a hard line on federal spending, which lawmakers are bracing to do battle over in January. When they return from the holidays on Wednesday, lawmakers will begin trying to pass a federal budget in a fight likely to be linked to	True

Table 3.1: Examples of articles extracted from the dataset (Fake vs Real News)

The final dataset after merging and adding the **label** column contains the following information:

Class	**Number of Articles**
Fake News	23,481
Real News	21,417
Total	**44,898**

Table 3.2: Class distribution in the dataset after preprocessing

In order to visualize the distribution of fake news and real news articles, here is a chart that shows the number of articles in each category.

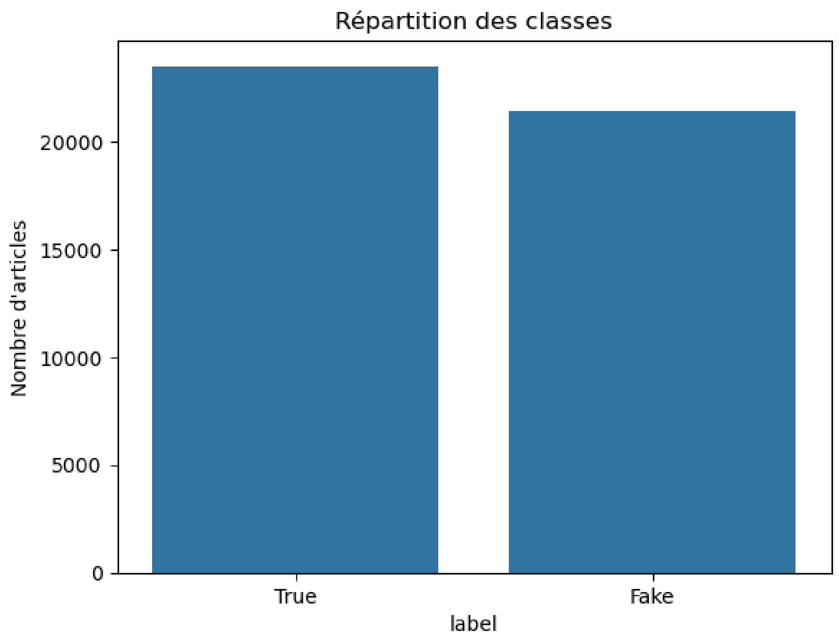

Figure 3.1: Distribution of articles by class (*Fake* vs *True*). The x-axis represents the two categories, and the y-axis represents the number of articles in each category.

The average text length, measured in number of words, is presented below for each article label (*Fake News* and *True News*):

Label	**Average Length (in words)**
0 (Fake News)	423.20
1 (True News)	385.64

Table 3.3: Average text length by label

The results indicate that*Fake News* articles are, on average, slightly longer than *True News* articles.

4. Preprocessing and Representation

Data preprocessing and text representation are essential steps in building a machine learning model for news article classification. The purpose of this section is to describe the various stages of the data preprocessing pipeline as well as the method used to transform texts into numerical vectors.

4.1 NLP Pipeline

Text preprocessing aims to clean and prepare the data for training the model. It was carried out through several steps:

- **HTML cleaning, punctuation removal, and lowercasing**: News articles often include HTML tags or other unwanted elements. These were removed using libraries such as BeautifulSoup. Additionally, all punctuation was stripped, and the texts were converted to lowercase to standardize the data and avoid unnecessary distinctions between uppercase and lowercase characters.
- **Stopwords**: Stopwords are frequent and non-informative words (such as "the", "and", "of") that were removed from all texts. These words are typically considered unhelpful in natural language processing tasks as they do not add meaningful context to the content. The stopword list was provided by the `NLTK` library.
- **Lemmatization**: Lemmatization involves reducing words to their canonical form (or lemma). For example, "eating" would be converted to "eat". This step reduces word variability in terms of form and thus improves the performance of machine learning models. For this task, the `spaCy` library was used.
- **Tokenization**: Tokenization is the process of breaking the text into smaller units called tokens. Each token can be a word, subword, or character depending on the tokenization method. In this project, tokenization was used to split the texts into individual words, facilitating their analysis and conversion into numerical vectors.

4.2 Word Embedding

Once the texts have been preprocessed, it is necessary to convert them into numerical vectors that a machine learning model can understand. To do this, we used the *word embeddings* technique, which represents each word as a dense vector in a vector space. In this project, the *Word2Vec* model was used to create word embeddings.

- **Word2Vec (trained on the project data)**: The Word2Vec model is a word representation technique that learns embeddings based on the contexts in which words appear. We trained a Word2Vec model on our project dataset to obtain vector representations specific to the fake news domain. The model was trained on a corpus consisting of the news articles, which had been processed using the NLP pipeline described above.
- **Word2Vec Model Parameters**: Several hyperparameters were defined when training the Word2Vec model to optimize the vector representations:

- `vector_size = 300`: The dimension of the word vectors is set to 300. This parameter defines the size of the vector space in which each word is represented.
- `window = 10`: The context window size is set to 10 words. This means the model considers the 10 words before and after a given word in the text to learn contextual relationships.
- `min_count = 3`: This parameter specifies that a word must appear at least 3 times in the corpus to be considered during training. This helps eliminate rare or low-frequency words that do not add significant value.
- `epochs = 20`: The number of training epochs is set to 20. This means the model will pass through the data 20 times to fine-tune the word representations.

These parameters were chosen after several experiments to ensure a balance between embedding quality and training efficiency.

4.3 Embedding Visualization with t-SNE

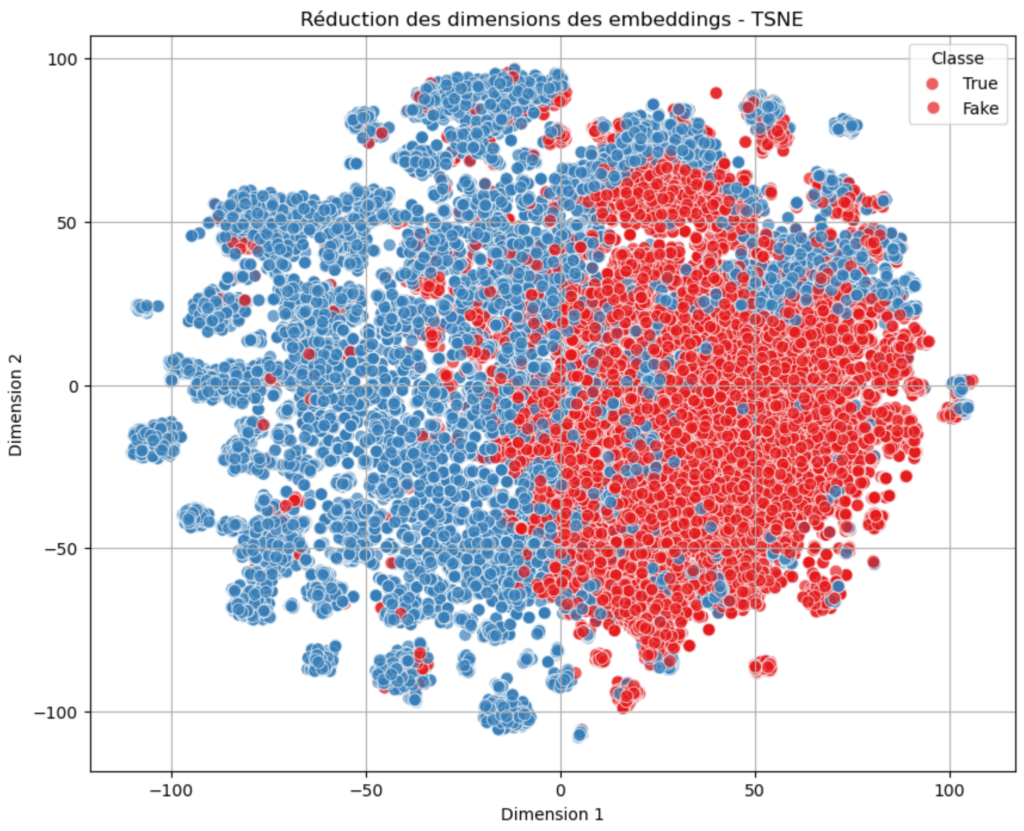

Figure 4.1: 2D Projection of Document Embeddings (Word2Vec) using t-SNE. Each point represents an article, colored according to its class (Fake or True).

2D Projection of Word2Vec Embeddings

This image represents a 2D visualization of the embeddings generated by a Word2Vec model, reduced to two dimensions using the *t-SNE (t-Distributed Stochastic Neighbor Embedding)* algorithm. Here are the key points to note:

- **Data Representation**:
  - **X-axis (Dimension 1)** and **Y-axis (Dimension 2)**:
    These axes correspond to the two main dimensions after dimensionality reduction. Although they are not directly interpretable, their layout reflects the similarities between points.
  - **Colored Points**:
    Each point represents a document (an article), with a different color according to its class (*Fake* or *True*). The legend displayed at the top right helps distinguish between these categories.
- **Cluster Interpretation**:
  - If points of the same color tend to cluster together, it suggests that the Word2Vec model has captured distinctive linguistic patterns between the classes.
  - Conversely, a significant mix of colors would indicate that the embeddings do not allow for a clear separation between categories.

5. Algorithm Design and Implementation

5.1 Data Splitting

After extracting and preprocessing the embeddings, the dataset was divided into three subsets: training, validation, and test. This split was performed using the `train_test_split` function from `scikit-learn`, following a standard 64%-16%-20% distribution. The table below summarizes this division:

Set	**Size**	**Percentage**
Training	28,735	80 %
Validation	7,184	10 %
Test	8,979	10 %
Total	44,898	100 %

Table 5.1: Data distribution across training, validation, and test sets.

Model: Bidirectional LSTM Network

The main model used in this project is a recurrent neural network of type LSTM (Long Short-Term Memory), designed to capture sequential and contextual dependencies in the vector representations generated by Word2Vec. The architecture details are as follows:

- **Input**: Word2Vec vector of size 300
- **Reshape**: $(300,) \to (300, 1)$
- **Bidirectional LSTM Layer (1)**: 128 units, `return_sequences=True`
- **Dropout**: 0.5
- **Batch Normalization**
- **Bidirectional LSTM Layer (2)**: 64 units, `return_sequences=False`
- **Dropout**: 0.5
- **Dense**: 64 units, `ReLU` activation
- **Dropout**: 0.3
- **Output (Dense)**: 1 neuron, `sigmoid` activation (for binary classification)

Training Parameters

- Optimizer: `Adam`
- Loss function: `binary_crossentropy`
- Metric: `accuracy`
- Epochs: 15 (with **early stopping** on the validation set)

- Batch size: 32

The model was trained using an *EarlyStopping* mechanism, which stops training when the validation loss does not improve for 3 consecutive epochs. This helps prevent overfitting.

Platform and Execution Environment

The project was developed in Python using the following libraries:

- `TensorFlow / Keras` for building and training the model
- `scikit-learn` for data handling and performance evaluation
- `NumPy, Pandas` for data manipulation

Training was conducted on a local machine equipped with:

- AMD : CPU
- RAM: 16 GB
- OS: Ubuntu 22.04

Model Saving

Once trained, the model was saved in `.keras` format in the `trained_models/` folder to allow for future reuse or deployment.

```
lstm_model.save("trained_models/lstm_model.keras")
```

6. Model Evaluation

Once the model is trained, it is essential to evaluate its performance on an independent test set to ensure its generalization capability. For this purpose, several **standard binary classification metrics** were used: **accuracy**, **precision**, **recall**, and **F1-score**.

6.1 Evaluation Metrics Obtained

Metric	**Value (%)**
Accuracy	93.91
Precision	92.93
Recall	96.22
F1-score	94.55

Table 6.1: Performance of the LSTM model on the test set

- **Accuracy** (overall correctness): Indicates that **93.91%** of the model's predictions are correct.
- **Precision**: Among the articles predicted as *real news*, **92.93%** were actually authentic. High precision reduces the risk of *false positives* (fake news wrongly classified as true).
- **Recall**: The model correctly identifies **96.22%** of real news. This means very few real articles are missed.
- **F1-score**: Combines precision and recall. With a value of **94.55%**, it reflects a good balance between both.

These results suggest that the **LSTM** model trained on Word2Vec vectors generalizes well, with excellent ability to distinguish between *fake news* and real news, even when texts are ambiguous or stylistically similar.

6.2 Confusion Matrix

The confusion matrix is a fundamental tool for evaluating the performance of a classification model. It provides a detailed view of the agreement (or disagreement) between the model's predictions and

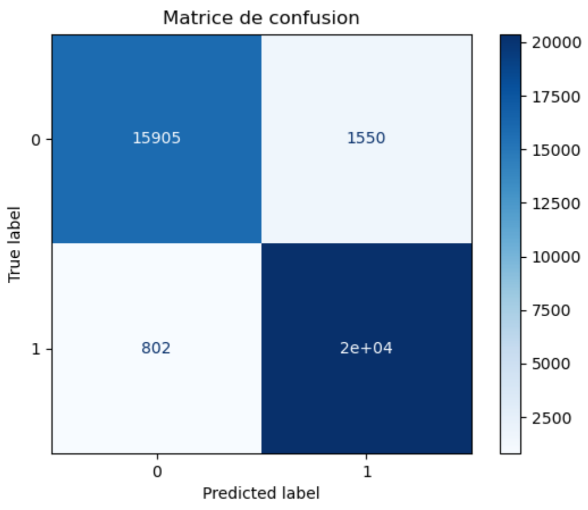

Figure 6.1: Confusion matrix of the LSTM model on the test set

Structure of the Matrix

- **Y-Axis (True Label)**: Actual labels from the data (**0** = **Fake**, **1** = **True**)
- **X-Axis (Predicted Label)**: Labels predicted by the model (**0** = **Fake**, **1** = **True**)

Interpretation of the Values (Approximate)

- **True Negatives (TN)**: ∼15,005 — The model correctly identified the *fake news*.
- **False Positives (FP)**: ∼802 — *Fake news* were incorrectly classified as true.
- **False Negatives (FN)**: ∼1,550 — Real news was incorrectly predicted as fake.
- **True Positives (TP)**: ∼20,000 — The model correctly identified the *real news*.

Analysis

- **Class Imbalance**: The "True" class (real news) appears to be more prevalent in the test set (20,000 TP vs. 15,000 TN), which may bias the model toward predicting real news more frequently.
- **Asymmetric Error**: The number of *false negatives* is noticeably high. This means the model sometimes rejects real news as fake, which could be problematic in a journalistic fact-checking context.
- **Overall satisfactory performance**, but adjustments such as better class weighting or data balancing could improve results for the minority class.

6.3 Performance Curve Analysis

6.3.1 ROC Curve (Receiver Operating Characteristic)

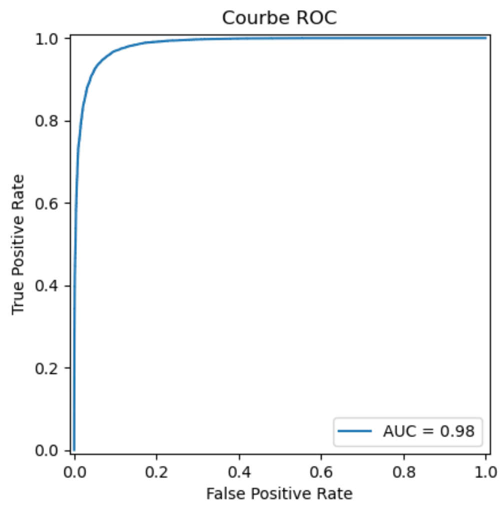

Figure 6.2: ROC Curve of the LSTM Model

The ROC curve evaluates the model's ability to distinguish between the two classes (fake and real news) by plotting the true positive rate (sensitivity) against the false positive rate.

- The curve is very close to the top-left corner, indicating excellent performance.
- The area under the curve (AUC) is **0.98**, which means that in 98% of cases, the model correctly ranks a real news article higher than a fake one in probabilistic terms.

Interpretation: An AUC close to 1 is a strong indicator of robustness for a binary classification model. It shows that the model has a high discriminatory power between the two classes.

6.3.2 Precision–Recall Curve

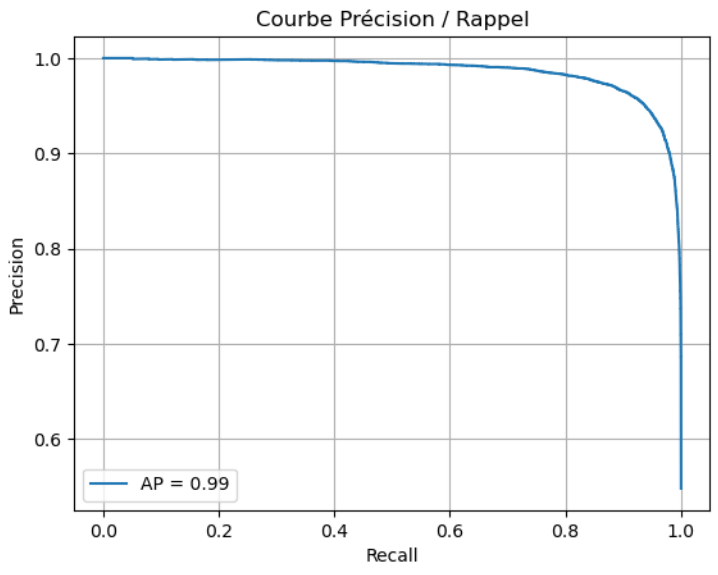

Figure 6.3: Precision-Recall Curve of the LSTM Model

This curve plots the **precision** (the proportion of true positives among all predicted positives) against the **recall** (the proportion of true positives that were correctly detected).

- The curve remains very high across almost the entire range, indicating that the model maintains high precision even at elevated levels of recall.
- The area under the curve (Average Precision, AP) is **0.99**, demonstrating that the model is extremely reliable at identifying real news among all positive predictions.

Conclusion: This curve is particularly useful in imbalanced scenarios or when false positives carry a high cost. Here, it confirms the stability and reliability of the LSTM model used for classification.

7. Error Analysis

Despite the model's strong performance, some classification errors remain. This section aims to identify and analyze the possible sources of these errors using several examples and interpretability tools.

7.1 Examples of Classification Errors

ID	Texte (extrait)	Vérité	Prédit
10559	*turn paid internet service quite yet job contract...*	Fake (0)	True (1)
33797	*new york reuters april reuters photographer zo...*	True (1)	Fake (0)
20	*la vega republican presidential nominee donald...*	True (1)	Fake (0)
39290	*reported line food shortage thing like toilet ...*	Fake (0)	True (1)
21114	*environmental protection agency looking hire d...*	Fake (0)	True (1)

Table 7.1: Examples of Classification Errors (False Positives and False Negatives)

7.2 Local Explanation with LIME

To better understand these errors, the LIME tool was used to interpret the predictions on certain documents.

- Some dimensions of the embeddings (e.g., `dim_7`, `dim_286`, etc.) have a positive or negative influence on the prediction.
- Example of interpretation: `dim_7 <= 0.10` increases the probability of the "Fake" class, while `dim_16 > 0.10` decreases it.

Limit: The explanation is complex to interpret directly since the dimensions are not semantically labeled. A projection or attention on the words would have been more readable.

7.2.1 SHAP Failure

The SHAP tool failed to generate an explanation because the default configuration (`max_evals=500`) was insufficient:

```
SHAP failed:  max_evals=500 is too low for the Permutation explainer,
it must be at least 2 * num_features + 1 = 601
```

Possible solution: Increase the value of `max_evals` to 601 or more to obtain actionable SHAP explanations.

7.2.2 Hypotheses on Sources of Errors

- **Content ambiguity:** Some articles may have a style or keywords similar to both classes, which confuses the model.
- **Sensationalist headlines:** A shocking headline may be associated with real information (false negative) or the opposite.
- **Latent imbalance:** Although the classes are relatively balanced, the lexical content of "True" articles seems to dominate some representations.
- **Limitations of average embeddings:** The approach of averaging Word2Vec vectors does not always capture finer syntactic or contextual relationships.

8. Conclusion and Possible Improvements

8.1 General Conclusion

This project aimed to develop a binary classification model that distinguishes fake news from real news articles based on their textual content (title and body). By using an NLP preprocessing pipeline combined with a Word2Vec model trained on the dataset, followed by a bidirectional LSTM network, we achieved remarkable performance:

- **Accuracy:** 93.91%
- **F1-score:** 94.55%
- **AUC (ROC):** 0.98
- **AP (Precision/Recall):** 0.99

These results demonstrate that the model is capable of learning discriminative and robust representations, despite the inherent complexity of the task (short texts, ambiguous vocabulary, inter-class similarity, etc.).

8.1.1 Identified Limitations

Despite these good results, several limitations should be noted:

- **Classification errors on ambiguous cases:** The model occasionally makes mistakes on texts that are either vague or ambiguous, especially when they use a neutral or sensationalist tone.
- **Domain specificity of the data:** The dataset used is primarily composed of articles related to politics. This limits the model's ability to generalize to other domains (health, science, economics, etc.), making its scope less universal for fake news detection in a broader context.
- **Risk of overfitting:** While the model achieves high accuracy on the training data, a slight decrease in performance on validation and test data may indicate overfitting. This suggests the model adapts too specifically to the training data and loses its generalization ability.
- **Limited interpretability:** The Word2Vec embeddings and LSTM model do not allow for direct understanding of the model's decisions.
- **Use of vector averaging:** This method loses word order and does not capture syntactic relationships effectively.

8.1.2 Possible Improvements

Here are some suggestions for improving the model's performance and interpretability:

- **Use a pre-trained model like BERT or RoBERTa** that can better contextualize words within sentences.

- **Implement an attention mechanism** to allow the model to focus on the most important parts of the text.
- **Refine text representation**: use contextual embeddings, or techniques like TF-IDF combined with explainable models.
- **In-depth error analysis with SHAP or LIME** on more examples to refine interpretation and correct any potential biases.
- **Data augmentation**: generate paraphrased texts to balance and diversify the training data.