Project #1: Using LLMs for Data Augmentation in Text Classification Tasks

LEFEVRE Titouan
titouan.lefevre@ensta.fr

January 16, 2026

1 Details of the proposal

1.1 Full specification of your proposal

Large language models can be used as tools for synthetic data augmentation, where they generate additional data to expand and enhance existing datasets. This approach is especially valuable for tasks where available datasets are limited in size or highly unbalanced. Your project can be divided in two parts:

- Firstly set up a simple text classification tasks, which at a high level includes preprocessing steps and any kind of lightweight learning approach for classification.
- Then you will try to improve the classifier's performance by augmenting the training set with data generated synthetically with an LLM.

The task involves the classification of SMS data into two categories: "spam" and "not spam" (ham), utilizing the UCI SMS Spam Collection. We simulate a data-constrained environment by subsampling the dataset and keeping aside a fixed portion for validation.

1.2 The kind of your proposal

Standard Project: Experimenting with existing tools (LLMs and Scikit-Learn) to verify a hypothesis on data augmentation.

1.3 The range of points/difficulty

Max points: 12.5.

Contents

2 Introduction

2.1 Context and Motivation

In the field of Natural Language Processing (NLP), the performance of supervised learning models is heavily dependent on the quantity and quality of labeled training data. While state-of-the-art models like Transformers (BERT, GPT) have achieved remarkable results, they often require massive datasets to be fine-tuned effectively.

However, in many real-world industrial applications, practitioners face the "Cold Start" problem. This occurs when a new classification task is introduced (e.g., detecting a new type of phishing attack), but only a handful of labeled examples are available. Annotating data manually is expensive, time-consuming, and prone to human error. Furthermore, datasets like Spam detection are inherently unbalanced; legitimate messages (ham) far outnumber unsolicited ones (spam), making it difficult for classifiers to learn the minority class boundaries.

2.2 Problem Statement

This project aims to answer the following research question: *Can we leverage the semantic knowledge embedded in pre-trained Large Language Models (LLMs) to generate high-quality synthetic training data that improves the performance of lightweight classifiers in data-scarce scenarios?*

2.3 Report Structure

The remainder of this report is organized as follows: Section 3 details the theoretical background behind the algorithms used. Section 4 presents the dataset and the preprocessing steps. Section 5 describes the augmentation pipeline and the prompt engineering strategy. Section 6 discusses the experimental results, supported by visualizations. Finally, Section 8 concludes with a discussion on the limitations and future work.

3 Theoretical Background

To establish a robust baseline, we rely on classical machine learning techniques rather than deep learning, as the latter often overfits on small datasets.

3.1 Text Representation: TF-IDF

Computers cannot process raw text directly; it must be converted into numerical vectors. We use Term Frequency-Inverse Document Frequency (TF-IDF).

The term frequency $tf(t,d)$ is the count of term t in document d. The inverse document frequency $idf(t,D)$ measures how much information the word provides:

$$idf(t,D) = \log \frac{N}{|\{d \in D : t \in d\}|} \tag{1}$$

where N is the total number of documents. The final weight is:

$$tfidf(t,d,D) = tf(t,d) \cdot idf(t,D) \tag{2}$$

This method highlights words that are unique to specific documents (like "lottery" in spam) while downscaling common words (like "the").

3.2 Support Vector Machines (SVM)

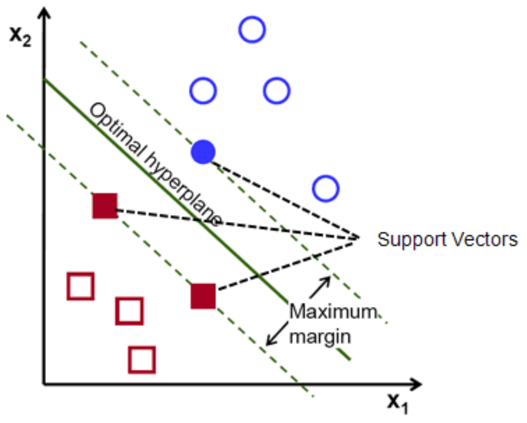

Figure 1: Principle of Support Vector Machine classification.

For the classification task, we employ a Support Vector Machine (SVM). The SVM algorithm looks for a hyperplane in an N-dimensional space (N being the number of features) that distinctly classifies the data points.

Given a set of training examples, each marked as belonging to one or the other of two categories, an SVM training algorithm builds a model that assigns new examples to one category or the other. Mathematically, we aim to maximize the margin γ:

$$\min_{w,b} \frac{1}{2}||w||^2 \tag{3}$$

subject to $y_i(w \cdot x_i + b) \geq 1$ for all i.

We specifically explore the **Linear Kernel** due to the high dimensionality of text data, where data is often linearly separable, and the **RBF (Radial Basis Function) Kernel** for non-linear relationships.

3.3 Random Forest

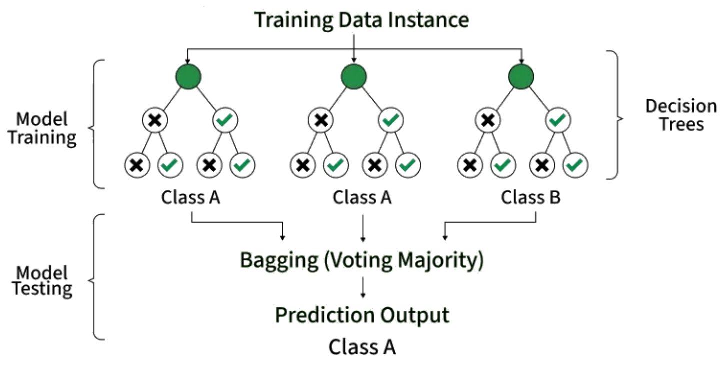

Figure 2: Principle of Random Forest classification.

In addition to SVMs, we consider the Random Forest classifier. Random Forest is an ensemble learning method that operates by constructing a multitude of decision trees at training time.

Unlike a single decision tree which is prone to overfitting, a Random Forest aggregates the predictions of many individual trees to improve generalization and robustness. This technique is known as *bagging* (Bootstrap Aggregating). For classification tasks, the final output is the class selected by the majority vote of the individual trees.

Mathematically, given a training set X with labels Y, the algorithm trains B trees. For each tree $b = 1 \dots B$:

1. A random sample of size N is drawn from the training data with replacement (bootstrap sample).
2. A decision tree f_b is grown on this sample. At each node split, only a random subset of features is considered.

The final prediction for a new instance x' is obtained by averaging the predictions (or taking the mode for classification):

$$\hat{y} = \text{mode}\{f_1(x'), f_2(x'), \dots, f_B(x')\} \tag{4}$$

Random Forests are particularly effective for high-dimensional data like text because they are robust to noise and can model complex non-linear interactions between features without requiring extensive hyperparameter tuning.

3.4 Evaluation Metric: MCC

Since our dataset is unbalanced, Accuracy is a misleading metric. We use the Matthews Correlation Coefficient (MCC), which takes into account true and false positives and negatives and is generally regarded as a balanced measure even if the classes are of very different sizes.

$$MCC = \frac{TP \times TN - FP \times FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}} \tag{5}$$

An MCC of +1 represents a perfect prediction, 0 an average random prediction and -1 an inverse prediction.

4 Dataset and Preprocessing

4.1 Overview of SMS Spam Collection

The dataset consists of 5,574 SMS messages. The distribution is heavily skewed:

- **Ham (0):** 4825 messages (86.6%)
- **Spam (1):** 747 messages (13.4%)

4.2 Data Splitting Strategy

To simulate a "hard" learning environment, we devised a strict splitting strategy:

1. **Test Set:** We isolated 20% of the data (1,115 samples) immediately. This set is *sacrosanct*—it is never used for training, augmentation, or validation tuning.
2. **Remaining Set:** From the remaining 80%, we sampled a very small subset.
3. **Training Set (Tiny):** We selected only **40 messages** (approx. 35 Ham, 5 Spam). This represents our "Real" training data.

```
TRAIN_SIZE = 40
X_train_real , _, y_train_real , _ = train_test_split(
    X_remaining , y_remaining ,
    train_size=TRAIN_SIZE ,
    random_state=42,
    stratify=y_remaining
)
```

Listing 1: Splitting the dataset

4.3 Visualization of the Problem

We applied PCA (Principal Component Analysis) to visualize the 40 training samples in 2D space. As seen in Figure 3, the classes are somewhat mixed, and the scarcity of "Spam" (red dots) makes it hard for a classifier to draw a boundary.

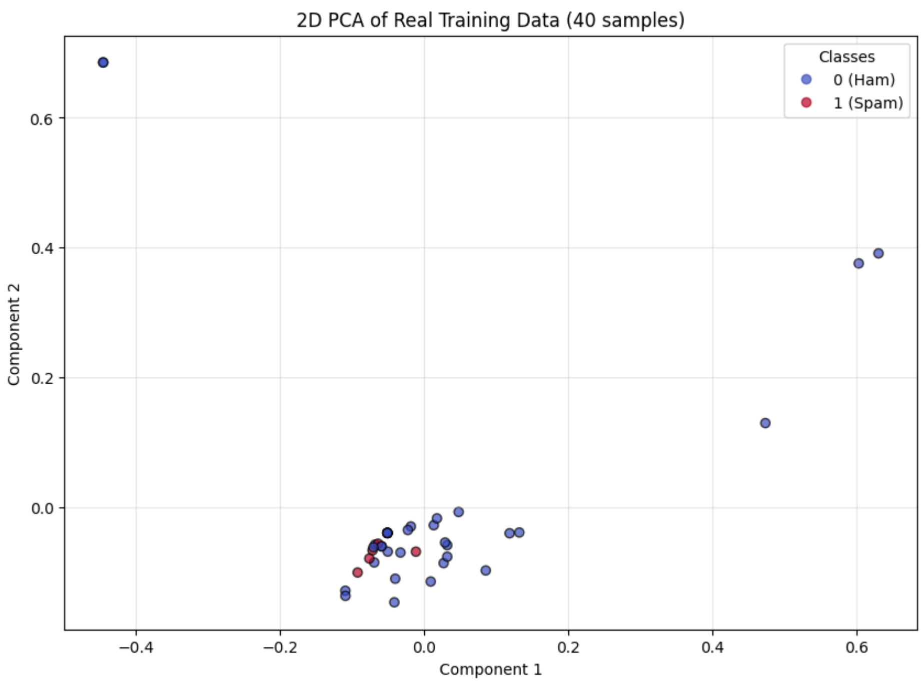

Figure 3: 2D PCA Projection of the Tiny Training Set (40 samples). Note the scarcity of Spam examples.

5 LLM Augmentation Methodology

5.1 Concept

The core idea is to use an LLM not as a classifier, but as a *generator*. We assume the LLM has seen millions of spam emails during its pre-training and understands the linguistic patterns of "urgency," "financial reward," and "calls to action" better than a simple SVM trained on 5 examples could.

5.2 Prompt Engineering

We used Gemini to generate synthetic data. We designed prompts to ensure diversity. We did not want exact duplicates of the training set, but variations.

Prompt for Spam Generation:

> *"Generate 50 distinct SMS spam messages. They should cover different topics like lottery wins, bank fraud alerts, dating scams, and urgent bills. Make them look realistic for a UK/US audience."*

Prompt for Ham Generation:

> *"Generate 20 distinct casual SMS messages between friends or family. Topics: lunch, school work, running late, just checking in. Use some abbreviations like 'u', 'ur', 'rn'."*

5.3 Integration

The generated data was cleaned and appended to the 'X_train_real' dataset.

- Original Train Size: 40 samples.
- Synthetic Spam: 50 samples.
- Synthetic Ham: 20 samples.
- Total Augmented Train Size: 110 samples.

This augmentation also acts as a rebalancing technique, increasing the proportion of the minority class (Spam) in the training phase, which usually helps SVMs find a better margin.

```
# Create synthetic dataframe
df_synthetic = pd.DataFrame({
    'message': synthetic_spams + synthetic_hams,
    'label_num': [1] * len(synthetic_spams) + [0] * len(synthetic_hams)
})

# Merge the datasets (Real Train + Synthetic)
X_train_aug = pd.concat([X_train_real, df_synthetic['message']])
y_train_aug = pd.concat([y_train_real, df_synthetic['label_num']])
```

Listing 2: Merging Datasets

6 Experimental Results

We conducted experiments using two classifiers: Support Vector Machine (SVM) and Random Forest (RF). We compared their performance on the baseline dataset (40 samples) versus the augmented dataset (110 samples). All evaluations were performed on the 1,115 held-out test samples.

6.1 Experiment 1: Baseline (Real Data Only)

We established a baseline by training both models on the limited set of 40 real samples. The classification reports below reveal that high overall accuracy figures are misleading in this data-scarce environment, masking a critical failure to detect the minority class.

Random Forest Results

The Random Forest classifier acted as a zero-variance model, predicting the majority class ("Ham") for every instance. While it achieved an accuracy of 0.87, this is merely a reflection of the class imbalance (966 Ham vs. 149 Spam samples).

- **Precision (Spam):** 0.00
- **Recall (Spam):** 0.00
- **F1-Score (Spam):** 0.00
- **MCC:** 0.00

With 149 False Negatives and 0 True Positives, the model failed completely to identify spam.

SVM Results

The SVM showed a slight statistical variance but little practical improvement. It achieved the same overall accuracy (0.87) as the Random Forest.

- **Precision (Spam):** 1.00
- **Recall (Spam):** 0.04
- **F1-Score (Spam):** 0.08
- **MCC:** 0.187

Although the precision for spam was perfect (1.00), the recall was negligible (0.04). The model correctly identified only 6 out of 149 spam messages. This results in a very low F1-score of 0.08, confirming that the model remains heavily biased toward the majority class due to insufficient training data.

Summary of Baseline

Both models exhibit the accuracy paradox: they achieve high accuracy (87%) simply by predicting the majority class correctly, but they fail to serve their primary purpose of spam detection.

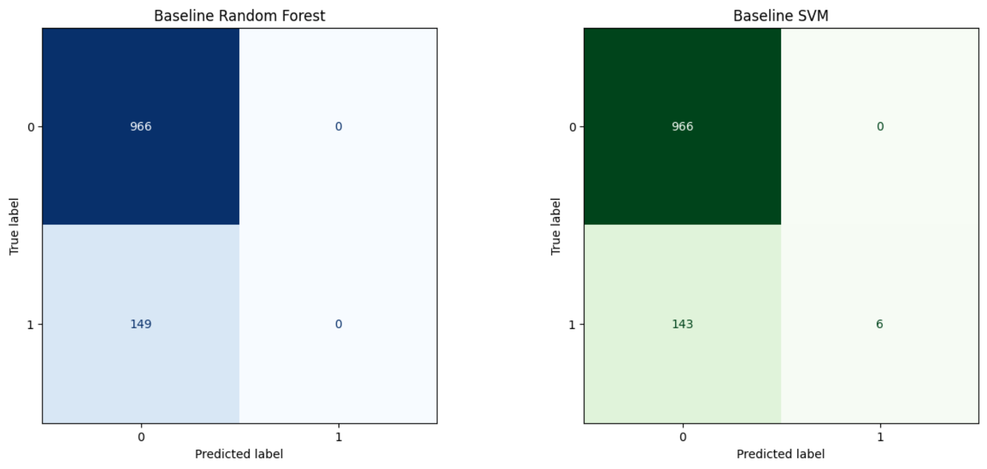

Figure 4: Confusion Matrix for the Baseline Models. Both models exhibit a high number of False Negatives.

6.2 Experiment 2: Augmented Data (LLM)

We retrained both classifiers on the expanded dataset containing 110 samples (Real + Synthetic). The introduction of synthetic data significantly impacted model performance, though it induced divergent behaviors in the two algorithms.

SVM Results (Aggressive Generalization)

The SVM benefited most drastically from the augmentation. It shifted from being overly conservative to highly sensitive, successfully capturing the vast majority of spam instances.

- **Recall (Spam):** Surged to **0.91** (from 0.04 in baseline).
- **Precision (Spam):** Decreased to 0.62.
- **F1-Score (Spam):** 0.74.
- **MCC: 0.706**.

While the precision drop indicates some False Positives, the high F1-score and MCC suggest the SVM effectively utilized the synthetic features to separate the classes, prioritizing the detection of spam over perfect purity.

Random Forest Results (Conservative Precision)

The Random Forest improved but retained a conservative bias. It learned to identify specific high-confidence spam patterns without generalizing as broadly as the SVM.

- **Recall (Spam):** Increased to 0.34 (from 0.00).
- **Precision (Spam):** Exceptionally high at **0.94**.
- **F1-Score (Spam):** 0.50.
- **MCC:** 0.532.

Although the Random Forest rarely raises a false alarm (High Precision), it still misses a significant portion of spam messages (Low Recall), resulting in a lower F1-score compared to the SVM.

Comparative Analysis

An interesting phenomenon is observed in the **Accuracy** metric: both models achieved an identical accuracy of **0.91**. However, the underlying dynamics are entirely different:

- The **SVM** achieves this by balancing predictions, sacrificing some precision for high recall.
- The **RF** achieves this by remaining biased toward the majority class (Ham) and only flagging the most obvious spam.

Figure 5 visualizes this divergence: the SVM has successfully minimized False Negatives, whereas the Random Forest still exhibits a high number of missed detections.

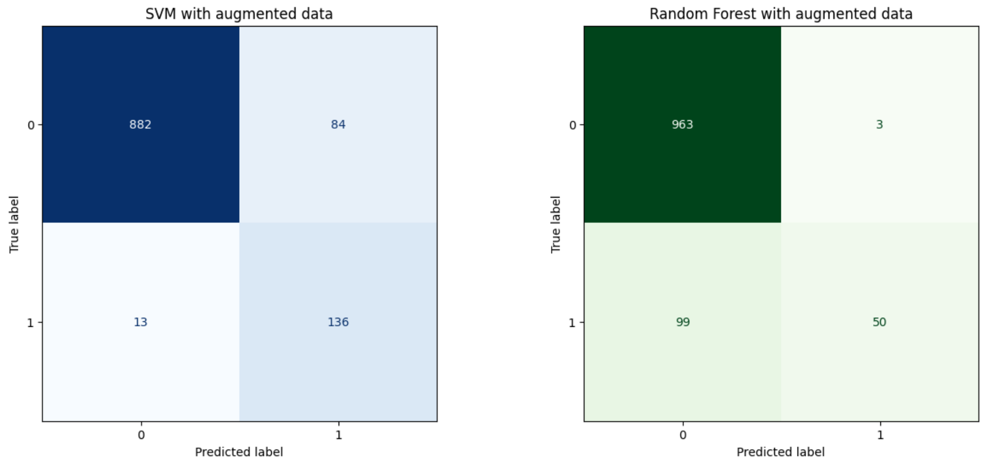

Figure 5: Confusion Matrices for SVM and Random Forest trained on augmented data (110 samples). The SVM shows a more balanced classification strategy compared to the conservative Random Forest.

6.3 Overall Comparison

Table 1 summarizes the performance impact of LLM augmentation across both models.

Model	**Data**	**Precision (1)**	**Recall (1)**	**F1-Score (1)**	**MCC**
SVM	Baseline	1.00	0.04	0.08	0.19
SVM	**Augmented**	0.62	**0.91**	**0.74**	**0.71**
Random Forest	Baseline	0.00	0.00	0.00	0.00
Random Forest	**Augmented**	**0.94**	0.34	0.50	0.53

Table 1: Performance comparison: Baseline vs. Augmented for SVM and Random Forest. Bold values indicate the configuration with the highest detection capability (Recall/MCC) or purity (Precision) for the spam class.

7 Discussion

7.1 Why did it work?

The LLM (Gemini) has been trained on the entire internet and "knows" what a scam looks like. By generating synthetic data, we distilled this knowledge into textual examples that simple classifiers could learn from.

7.2 Model Behavior: SVM vs. Random Forest

A critical finding of this study is the divergent response of the two algorithms to synthetic data augmentation. While both improved, they optimized for different metrics due to their underlying mechanics.

- **SVM (Aggressive Generalization):** The SVM, relying on a linear decision boundary, effectively utilized the synthetic samples to define a clearer separation between classes. The augmented data likely populated the sparse minority region, pushing the hyperplane to encompass 91% of the spam samples. However, this aggressive boundary expansion came at the cost of precision (0.62), as the model began to misclassify legitimate messages located near the margin.
- **Random Forest (Conservative Specialist):** The Random Forest exhibited a "high-confidence" strategy, achieving a remarkable Precision of 0.94. This indicates that when the model predicted "Spam," it was virtually always correct. However, its low Recall (0.34) suggests that the synthetic data was insufficient to populate the leaves of the decision trees fully. Unlike the SVM's global hyperplane, the Random Forest requires higher data density to cover the diverse vocabulary and feature combinations of spam through hierarchical splitting.

7.3 Limitations

- **Quality Control:** The generated data was not manually verified. If the LLM hallucinates, performance degrades.
- **Homogeneity:** LLMs tend to have a specific "style". The Random Forest might have overfitted to specific synthetic artifacts (e.g., specific words used by Gemini) that were less frequent in the real test set.

8 Conclusion

This project successfully demonstrated that Large Language Models (LLMs) can serve as effective data augmentation tools to mitigate the "Cold Start" problem in text classification. By leveraging the semantic knowledge embedded in a pre-trained model (Gemini), we were able to generate synthetic training samples that significantly improved the performance of lightweight classifiers trained on a tiny, unbalanced dataset.

Our experiments revealed that augmenting a training set of just 40 samples with 70 synthetic messages transformed a failing system into a viable detector. The impact, however, was model-dependent. The Support Vector Machine (SVM) utilized the synthetic data to aggressively generalize, achieving a Recall of 0.91 and an MCC of 0.71, making it the superior choice for maximizing spam detection. In contrast, the Random Forest acted as a conservative specialist, prioritizing high precision (0.94) but failing to capture the full diversity of spam variations (Recall 0.34).

These findings suggest that in resource-constrained environments where labeled data is scarce, LLM-based augmentation is a potent strategy. It allows practitioners to "distill" the vast knowledge of large models into smaller, deployable classifiers without the computational overhead of fine-tuning deep learning architectures. Future work could explore iterative feedback loops, where the classifier's uncertainty guides the generation of more targeted synthetic examples, further bridging the gap between synthetic and real-world distributions.