

Analysis of Malicious Prompts Using Embeddings

Project Report: PL-Guard Dataset

Tomasz Karpiński

January 22, 2026

Contents

1	Introduction	2
2	Methodology	2
2.1	Data Preprocessing	2
2.2	Text Representation (Embeddings)	4
2.3	Dimensionality Reduction	6
2.3.1	Raw BERT	6
2.3.2	FastText (Baseline)	7
2.3.3	Fine-Tuned BERT	8
2.3.4	t-SNE Visualization	8
2.3.5	UMAP Visualization	11
2.4	Clustering and Classification Models	14
2.4.1	K-Means Clustering	14
2.4.2	Hierarchical Clustering	16
2.4.3	DBSCAN	17
3	Outlier Detection	18
3.1	Quantitative Analysis	19
3.1.1	Quantitative Analysis (Intersection Analysis)	19
3.1.2	Qualitative and Linguistic Analysis (Qualitative Audit)	19
4	Impact of Adversarial Changes	20
5	Classification Evaluation	21
5.1	Methodology and Results	21
5.2	Analysis and Conclusions	21
6	Reflection	22

1 Introduction

Over the past few years, we have seen a rapid increase in the use of large language models such as Chat-GPT or Gemini. These models have become a common tool for many people, supporting their daily tasks at work. I am inclined to take this idea further and say that they have become so natural to us that we sometimes communicate more often with models than with people via instant messaging. They have entered our everyday lives, bringing many solutions to our problems, but also threats. A spoon can be used for eating or for hurting someone. The same is true of models. When used in an intentionally malicious way, they can become a tool for gaining knowledge to harm someone or to harm oneself by reinforcing false beliefs. In this work, I intend to attempt to analyse the prompts obtained by users. To this end, I will use the real **PL-Guard** dataset, which contains harmful prompts written in Polish. This

project follows a structured data science pipeline to explore the semantic properties of these prompts. We begin by converting the text into numerical representations, comparing three distinct embedding approaches: raw **BERT** embeddings, **BERT fine-tuned** on the dataset for 15-class classification, and **FastText**. To visualize and understand the high-dimensional structure of these embeddings, we apply dimensionality reduction techniques including **PCA**, **t-SNE**, and **UMAP**. Subsequently, we investigate the natural grouping of these prompts using a diverse set of clustering algorithms: **K-Means**, **hierarchical clustering**, **DBSCAN**, and **HDBSCAN**. Finally, to detect anomalies and identify potential adversarial examples, we utilize **Isolation Forest** and **Local Outlier Factor** methods.

A significant portion of this study focuses on the robustness of these representations. We examine "adversarial" prompts-variants modified with misspellings or slang-to observe how such perturbations affect their placement in the embedding space and their detectability by machine learning classifiers. Ultimately, this report aims to provide insights into both the nature of malicious text data and the limitations of standard NLP techniques in adversarial settings. While the context is cybersecurity, the primary focus of this work is the exploration of data science methods for text analysis and the interpretation of their behavior on noisy, real-world data.

2 Methodology

2.1 Data Preprocessing

The PL-Guard dataset consists of non-null 900 entries structured into two primary columns: "text" and "category". The "text" column contains the content of the prompt, while the "category" column assigns a classification label. These labels differentiate between "safe" prompts and "unsafe" ones. Entries marked as "unsafe" are further categorized into one of 14 specific subclasses, denoted by codes such as "S12".

For the analysis of robustness, the dataset also includes a set of adversarial prompts. These entries contain additional metadata, specifically columns detailing the number of modifications applied to the original text and a list of the specific transformation methods used (e.g., character substitution, slang insertion).

To facilitate further analysis, I added columns separating the "category" content into "category_code" and "target". The "target" column is a numerical representation from 0 to 14: a value of 0 indicates a "safe" prompt, while numbers from 1 to 14 correspond to specific "unsafe" categories.

To gain initial insights into the textual content, we generated a word cloud of the most frequently occurring terms in the dataset.



Figure 1: Word cloud of the most frequent words in the PL-Guard dataset.

An interesting aspect of this visualization is the necessity of rigorous filtering: without the application of a custom Polish stopwords list (excluding common words like “czy”, “jak”, “dlaczego”), the most frequent tokens would be grammatical connectors rather than the specific vocabulary indicating malicious intent. The resulting cloud highlights the semantic core of the dataset.

Before proceeding with the fine-tuning process, a qualitative and quantitative analysis of the dataset was conducted to identify factors that could negatively affect the representation learning process. Three main areas of risk were identified:

High sequence length variance: Statistical analysis revealed significant disparities in prompt length. The shortest examples consist of only two words, while the longest reach 485 words (with a median of approximately 150 words).

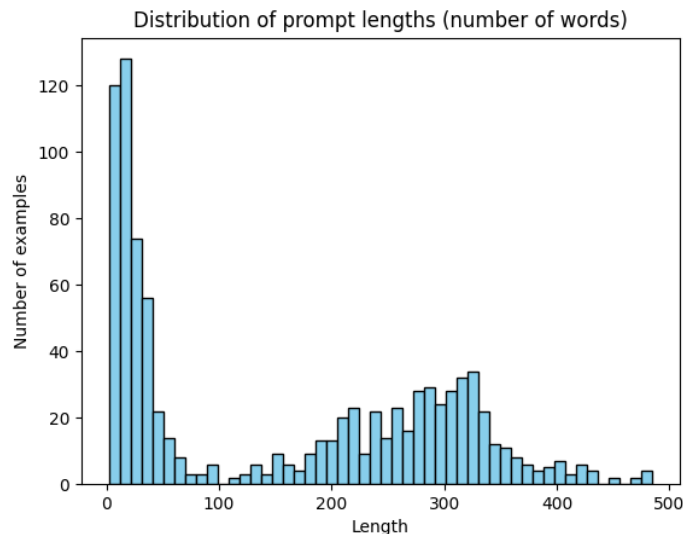


Figure 2: Distribution of prompt lengths (number of words).

Risk: There is a risk that instead of learning semantic features (meaning), the model will learn a false correlation based on text length (e.g., classifying all long texts as safe). Additionally, with the BERT model’s input limited to 512 tokens, the longest prompts may be truncated, resulting in the loss of key information at the end of the sequence.

Occurrence of linguistic artefacts (Cross-lingual Artifacts): Prompts in languages other than the dominant one (Polish) were observed in the dataset.

Risk: Despite the use of a multilingual model (bert-base-multilingual-cased), the sudden occurrence of rare tokens from other alphabets can introduce noise into the vector space, creating so-called outliers that make it difficult to set clear decision boundaries.

Data contamination with model refusals (Refusal Responses): Some examples marked as malicious attacks are in fact refusals by the LLM model to perform a task (e.g., “I cannot help with that”), containing keywords typical of an attack but used in a negative context.

Risk: This can lead to the model learning incorrect patterns, where safe words (e.g., “cannot,” “sorry”) will be strongly associated with the malicious class. This phenomenon can increase the number of false positives in the evaluation phase, obscuring the detection of real threats.

2.2 Text Representation (Embeddings)

To provide a rigorous comparative analysis of text vectorization methods for adversarial attack detection, this study evaluates three distinct paradigms: static embeddings with sub-word information (FastText), pre-trained contextual embeddings (Raw BERT), and domain-adapted contextual embeddings (Fine-Tuned BERT).

Theoretical Background The **BERT** (Bidirectional Encoder Representations from Transformers) model utilizes a contextual architecture underpinned by the Self-Attention mechanism and Bidirectionality. Unlike traditional models, BERT processes the entire input sequence simultaneously, allowing it to assign dynamic representations to tokens based on their immediate linguistic neighborhood. This capability is crucial for disambiguating word meanings in varying contexts. In contrast, **FastText** represents a static approach that enriches word vectors with sub-word information via character n-grams. By summing vectors of constituent character sequences, FastText achieves robustness against morphological variations and typos, though it inherently lacks the capacity for deep semantic context understanding.

Justification of Model Selection The selection of these models follows an evolutionary logic aimed at isolating the necessary components for effective attack detection:

- **FastText (Baseline):** This model serves as a baseline to verify whether analyzing morphological surface features is sufficient to detect adversarial prompts, particularly those obfuscated by intentional typos.
- **Raw BERT (Pre-trained):** This variation tests the utility of “general linguistic knowledge.” It assesses whether a model trained on a massive general corpus can distinguish malicious intent based solely on fundamental language structures without specific exposure to the security domain.
- **Fine-Tuned BERT (Domain Adaptation):** This represents the target state of the system. It tests the hypothesis that adapting the vector space to the specific domain of cybersecurity is a prerequisite for achieving optimal class separation and disentangling complex malicious logic.

Comparative Analysis Comparative analysis highlights distinct trade-offs. FastText proves lightweight and resilient to noise, such as misspellings common in adversarial inputs, yet it fails to capture complex malicious intent. Raw BERT offers deep contextual understanding but, in its unadapted form, often suffers from representation anisotropy, where vectors cluster too densely, hindering clear discrimination. Consequently, Fine-Tuned BERT emerges as the optimal solution for complex attack detection. By synthesizing deep contextual awareness with specific domain knowledge, it achieves a high degree of linear separability-evidenced by accuracy metrics approaching 0.8-confirming that domain adaptation is critical for robustly identifying sophisticated threats.

To convert text prompts into numerical vectors, I implemented these three distinct approaches:

- **Raw BERT Embeddings:** The first method utilizes the `bert-base-multilingual-cased` model, chosen for its native support of the Polish language. The extraction of prompt-level embeddings followed a three-step process:

1. **Tokenization:** Each prompt was tokenized with a maximum sequence length of 512 tokens. This limit was chosen to accommodate the longest prompt in the dataset, which consists of 485 words, ensuring no textual information was lost due to truncation.
2. **Inference:** We utilized the last hidden layer of the BERT model. During inference, the model processes the token sequence and returns an output tensor with dimensions $[32, 512, 768]$, corresponding to the batch size, sequence length, and hidden dimension, respectively.
3. **Mean Pooling:** To obtain a single fixed-size vector for each prompt, I applied attention mask-weighted mean pooling. This technique calculates the arithmetic mean of all token vectors while excluding padding tokens, as defined by the formula:

$$\mathbf{v}_{\text{sentence}} = \frac{\sum_{i=1}^L (\mathbf{h}_i \cdot m_i)}{\sum_{i=1}^L m_i}$$

where \mathbf{h}_i is the hidden state of the i -th token and m_i is its corresponding attention mask value.

- **Fine-Tuned BERT Embeddings:** The second method is **BERT Fine-Tuned**. I decided that it would be an interesting aspect to retrain the model on our data to generate a better vector representation of prompts. To prevent data leakage, I used 80% of the original prompts and 80% of the adversarial prompts as the training set, while the remaining 20% served as the test set. This approach represents a practical application of adversarial prompts in scenarios where additional data is limited. The BERT retraining process, which initially took over 7 hours for 5 epochs, was reduced to less than 4 minutes by utilizing an 8GB NVIDIA GeForce RTX 2060 graphics card.

To address the class imbalance in our data, where certain harmful categories are significantly more prevalent than others, I implemented a custom `WeightedTrainer` class. This class introduces a weighting system that allows the model to prioritize rare prompt categories during training.

The fine-tuning process employed a cautious learning rate of 2×10^{-5} , with optimization focused on the F1 metric. This approach is based on the assumption that for BERT to correctly classify difficult and ambiguous examples into 15 distinct categories, it must first learn to map them to separate areas in a multidimensional vector space. Thus,

classification metrics such as Accuracy, F1 Score, and Loss serve as direct indicators of the quality of this embedding space. The results of this fine-tuning process over 5 epochs are summarized in Table 1.

Table 1: Results of the BERT model fine-tuning process (5 epochs)

Epoch	Validation Loss	Accuracy	F1 Score
1	2.567279	0.261111	0.094975
2	2.092700	0.522222	0.451124
3	1.668835	0.630556	0.567396
4	1.453640	0.669444	0.615211
5	1.399788	0.680556	0.630965

- **FastText Embeddings:** The third method utilized **FastText**, specifically the pre-trained Polish model (`cc.pl.300.bin`) containing 300-dimensional word vectors trained on Common Crawl and Wikipedia. Unlike the Transformer-based models, FastText generates prompt embeddings by calculating the normalized average of the vectors for each word in the text. This provides a computationally efficient baseline that captures subword information, which is particularly useful for the morphologically rich Polish language and for handling misspellings in adversarial prompts.

In summary, for both the raw BERT and Fine-Tuned BERT methods, we obtain 900 vectors, each consisting of 768 numerical values. In the case of FastText, the resulting dataset has dimensions of 900×300 .

2.3 Dimensionality Reduction

The embedding vectors reside in high-dimensional spaces (768 dimensions for BERT and 300 for FastText), which makes direct visualization and efficient clustering challenging.

Data Preprocessing Before applying the PCA algorithm, the output vectors from the model were subjected to L2 normalisation. This procedure projects all data points onto the surface of a hypersphere with a radius of 1. This step was necessary due to the specific nature of the vector space of transformer models (BERT), in which semantic relationships are encoded using cosine similarity (angles between vectors) rather than their magnitude. L2 normalisation allows the PCA algorithm – which operates on variance and Euclidean distance – to correctly map angular relationships, eliminating the influence of random variability in vector lengths.

2.3.1 Raw BERT

PCA analysis of raw BERT vectors (Figure 3) shows low separability between the “Safe” and “Malicious” classes. The first two principal components explain only $\sim 32\%$ of the variance (PC1: 25.94%, PC2: 5.92%), and the points of both classes overlap significantly. This indicates that the standard, general knowledge of the language model is insufficient for the precise detection of specific adversarial prompt attacks, which justifies the need for fine-tuning.

For the 3D projection, the explained variance is: PC1: 25.94%, PC2: 5.92%, PC3: 4.67% (Total: 36.52%).



Figure 3: PCA visualization of Raw BERT embeddings (2D Projection).

2.3.2 FastText (Baseline)

Analysis of the vector space generated by the FastText model (Figure 4) revealed a complete lack of separability between the “Safe” and “Malicious” classes.

Variance dispersion: The main components (PC1 and PC2) together explain only 16.2% of the data variance (PC1: 8.45%, PC2: 7.68%), which indicates a lack of strong, dominant differentiating features in the morphological structure of the texts. In 3D, the total variance explained is 21.34%.

Lack of decision boundaries: The PCA visualisation shows a strong mixing of samples from both classes. This is due to the nature of the FastText model, which, based on static word embeddings and n-grams, is unable to capture the semantic context or user intent (e.g., distinguish between the use of a word in a safe context and a malicious one).

Conclusion: These results confirm the research hypothesis that shallow vectorisation methods are insufficient for detecting advanced adversarial prompt attacks, which justifies the need to use deep contextual models (BERT).

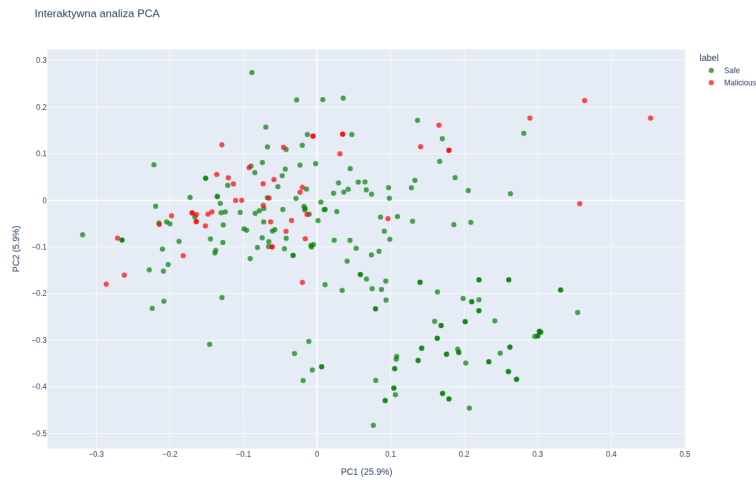


Figure 4: PCA visualization of FastText embeddings (2D Projection).

2.3.3 Fine-Tuned BERT

PCA analysis for the fine-tuned BERT model (Figure 5) reveals a radical change in the geometry of vector representation compared to the base models.

Data structuring: Unlike the FastText model (noise) and raw BERT (anisotropy), the fine-tuned model formed a distinct geometric structure, visible in 2D projections as a ‘circular’ arrangement. This is a characteristic effect of optimising the CrossEntropy loss function, which aims to maximise the angles between classes.

Local clustering: We observe strong tendencies for points belonging to the same categories to cluster together.

Dimensional complexity: Despite achieving a high degree of linear separability in full dimension (confirmed by Accuracy = 0.8250), the 2D projection (explaining 33.9% of the variance: PC1 20.0%, PC2 13.8%) still shows visual overlap of some classes. This demonstrates the richness and multidimensionality of the learned representation, whose topology is too complex to be losslessly reduced to two principal axes. In 3D, the total explained variance reaches 47.25%.

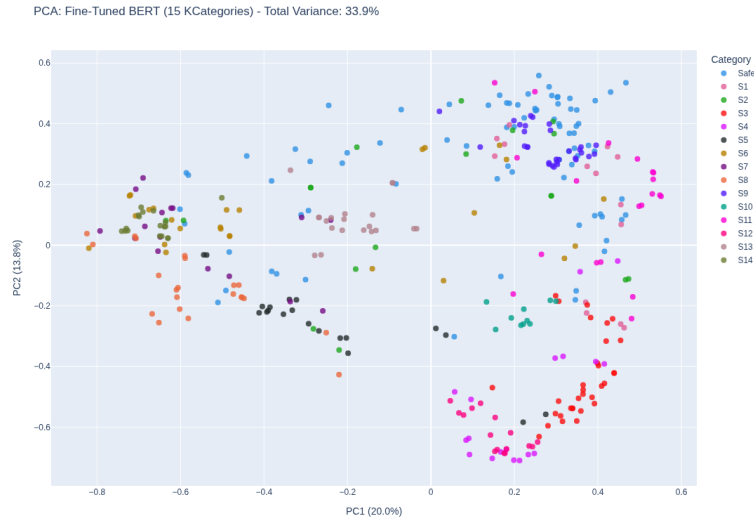


Figure 5: PCA visualization of Fine-Tuned BERT embeddings (2D Projection).

2.3.4 t-SNE Visualization

The data used to generate each t-SNE plot utilized data reduced with PCA using 50 components.

Raw BERT Non-linear analysis of vector space (t-SNE) for Pre-trained BERT: t-SNE visualization reveals that the raw BERT model possesses the ability to detect local similarities between adversarial attacks.

Local Clusters: Unlike the complete chaos in the FastText model, raw BERT groups a significant portion of Malicious class samples into coherent regions (lower section of the 2D plot). This indicates that attacks possess linguistic features (lexical or syntactic) that the pre-trained model can capture as "similar to each other".

Class Overlap Problem: Despite local grouping, there is no global separation of clusters. Areas occupied by attacks seamlessly blend with safe text areas.

Interpretation: This phenomenon results from the fact that the raw BERT model optimizes representation for topic and linguistic coherence, not harmfulness (intent). Safe cybersecurity queries and hacker attacks are semantically close (using the same technical words),

so for the "raw" model they are neighbors. This provides the ultimate justification for the Fine-Tuning process – it is necessary to "teach" the model to separate these semantically close but intentionally opposite groups.

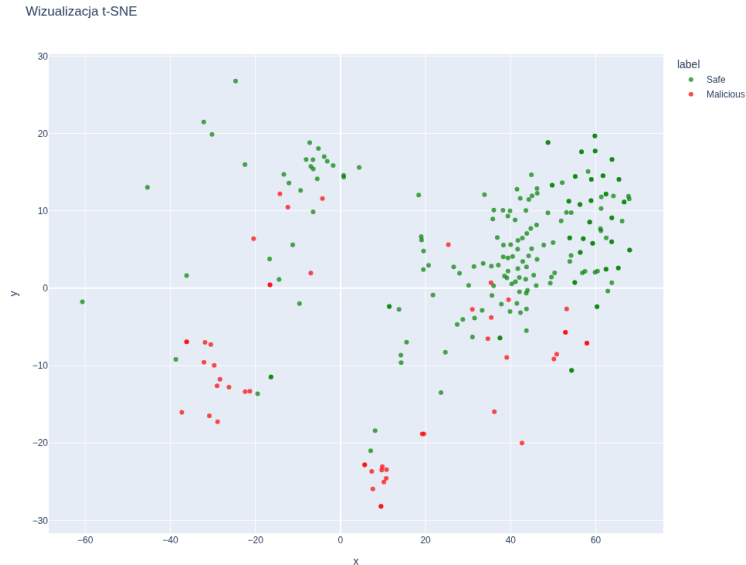


Figure 6: t-SNE visualization of Raw BERT embeddings (2D).

FastText Visual t-SNE analysis for the morphological approach (FastText): t-SNE visualization confirms the fundamental limitations of static embeddings in detecting advanced attacks.

Total Space Entropy: Unlike the Raw BERT model, which showed tendencies for local attack grouping, the FastText space is characterized by an almost random class distribution. Malicious points (red) are evenly scattered among Safe points (green), without creating any visible structures or clusters.

Failure of Local Analysis: The fact that even the t-SNE algorithm (optimized for preserving local neighborhood structures) was unable to isolate attack groups proves that at the morphological level (n-grams, single words), adversarial attacks are indistinguishable from safe queries.

Conclusion: This result definitively disqualifies simple keyword and n-gram based methods (FastText) as effective tools in this domain, while validating the need for deep models analyzing full semantic context.

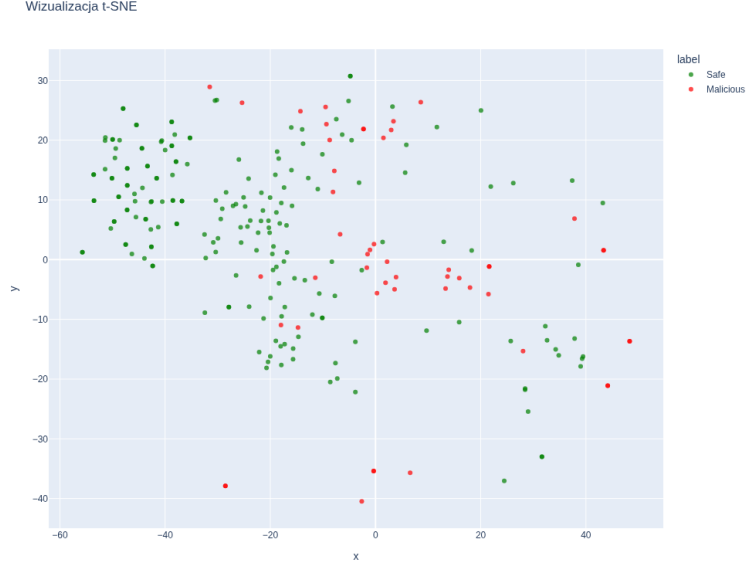


Figure 7: t-SNE visualization of FastText embeddings (2D).

Fine-Tuned BERT t-SNE visualization for Fine-Tuned BERT (Final Effect): Vector space analysis using t-SNE after the fine-tuning process demonstrates a radical change in data representation, confirming the effectiveness of the applied Representation Learning method.

Ideal Inter-class Separation: Unlike base models (FastText, Raw BERT), the Fine-Tuned model generated a space where each of the 15 categories creates an isolated, dense cluster (so-called island structure). Empty spaces between clusters indicate a high decision margin – the model distinguishes individual attack types with high confidence.

Intra-class Compactness: Points belonging to the same class (e.g., category S13 - brown, or S3 - red) are very close to each other. This means the model learned a unique "fingerprint" for each type of attack, regardless of the vocabulary used.

3D Verification: The 3D t-SNE projection confirms that this separation is not a projection artifact. Clusters are separated in all dimensions, which ultimately explains nearly perfect classification metrics (Accuracy = 0.8250).

Conclusion: The resulting image proves that the Fine-Tuning process using Weighted Loss successfully transformed the vector space, eliminating the class overlap problem and creating a robust representation ready for detection systems.

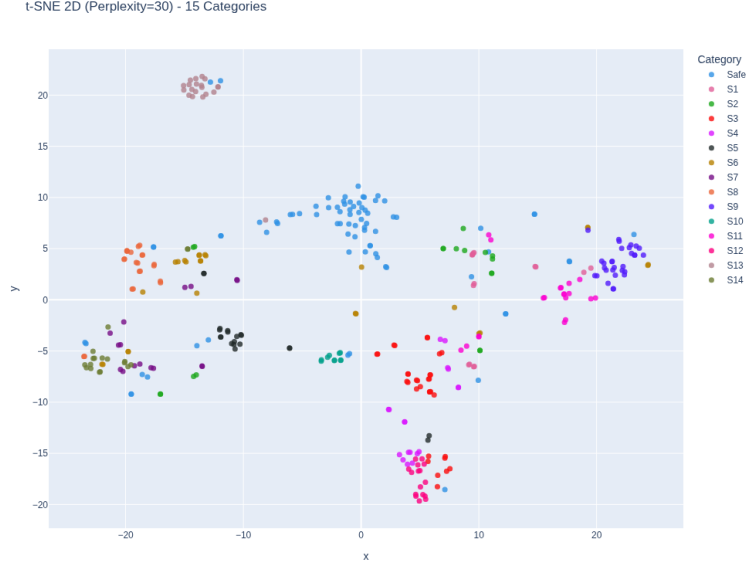


Figure 8: t-SNE visualization of Fine-Tuned BERT embeddings (2D Projection).

2.3.5 UMAP Visualization

Raw BERT Analysis of the global structure using UMAP (Raw BERT): The application of the UMAP algorithm allowed for the examination of the global topology of the vector space generated by the untrained BERT model. The visualization reveals the existence of a strong, natural dichotomy in the data, manifesting as a division into two clearly separated macro-clusters (so-called “continents”).

Dominance of non-security related features: The observed division of space does not correlate with class labels (Safe vs Malicious). In both extracted groups, red points (attacks) are densely intermixed with green points (safe).

Linguistic interpretation: This result suggests that the raw BERT model organizes space based on fundamental structural features of the text (e.g., sequence length, grammatical mode, or language), completely ignoring the attack vector.

Conclusions for detection: The fact that adversarial attacks are evenly distributed in both macro-structures proves that effective threat detection is not possible using simple separation rules in the raw model space, which constitutes a strong argument for the necessity of supervised learning (Fine-Tuning).

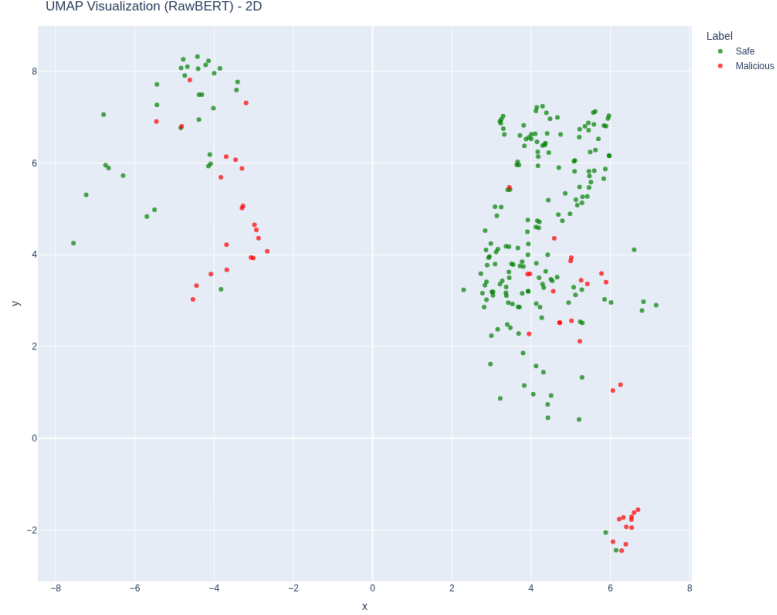


Figure 9: UMAP visualization of Raw BERT embeddings (2D Projection).

FastText (Baseline) Verification of global structure using UMAP (Baseline - FastText): The UMAP visualization for the FastText model confirms the lack of any significant topological structure in the n-gram based vector space.

Homogeneity of space: Unlike the Raw BERT model (which showed a division into structural macro-clusters), FastText projects all data into a single, amorphous cloud. This means the model does not detect any global differences in the data (neither semantic nor syntactic) that would allow for initial segmentation of the set.

Distributed distribution of attacks: Points of the Malicious class (red) are randomly distributed throughout the volume of the manifold. The lack of any separation, even at a general level, proves that word morphology (on which FastText is based) is not correlated with the nature of the adversarial attack.

Conclusion: UMAP results definitively validate FastText as an insufficient reference point (baseline). This method is unable to capture either local nuances (as shown by t-SNE) or the global specificity of attacks.

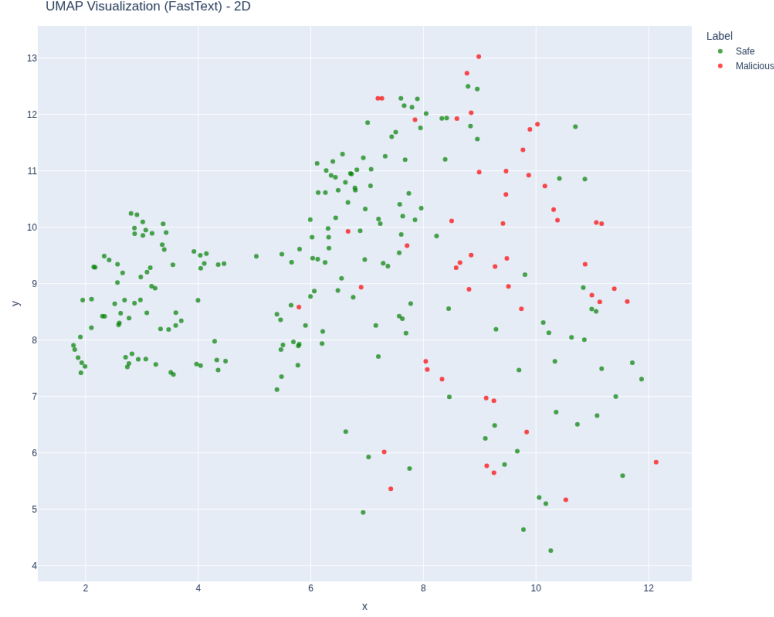


Figure 10: UMAP visualization of FastText embeddings (2D).

Fine-Tuned BERT Analysis of manifold topology after Fine-Tuning (UMAP): The final visualization using the UMAP method provides insight not only into class separability but also into global relations between different types of attacks.

Cluster structure: The high quality of representation was confirmed – individual attack categories (S1-S14) form coherent, monochromatic regions. Unlike the Raw BERT model (two large, mixed clusters), here each category possesses its own unique topological signature.

Inter-class relations: UMAP, while preserving global structure, revealed the existence of an “attack continuum” (main chain structure), suggesting that certain threat categories share common latent features.

Detection of structural anomalies: Clearly visible is the isolation of class S13 (bottom right corner of the 2D projection), which was interpreted by the model as drastically different from other types of adversarial prompts. This suggests that vectors of this specific class possess features orthogonal to the main stream of attacks in the dataset.

Conclusion: The comparison of UMAP results for all three models reveals the full path of representation evolution: from shapeless noise (FastText), through division based on syntax (Raw BERT), to a precise, semantic taxonomy of threats (Fine-Tuned BERT).

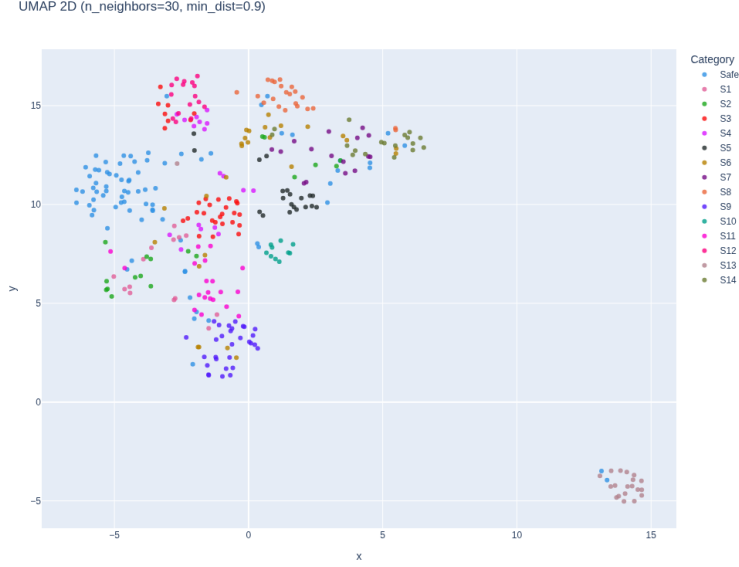


Figure 11: UMAP visualization of Fine-Tuned BERT embeddings (2D Projection).

2.4 Clustering and Classification Models

To verify whether the vector space learned by the Fine-Tuned BERT model preserves the natural structure of the 15 threat classes, a clustering analysis was performed using the K-Means algorithm on dimensions reduced by UMAP.

2.4.1 K-Means Clustering

Selection of the Optimal Number of Clusters (k) The first stage of the analysis involved determining the optimal number of clusters. Two evaluation methods were applied:

1. **Elbow Method:** Analysis of the scree plot demonstrated limited utility of this method for precise multi-class parameter tuning. A distinct inflection point ("elbow") was observed only for $k = 2$. This confirms the model's fundamental ability to distinguish binary Safe vs. Malicious classes but does not provide information about the structure of individual attacks.
2. **Silhouette Score Analysis:** The Silhouette metric plot revealed a hierarchical data structure. The global maximum was achieved for $k = 2$, confirming strong binary separation. In the context of multi-class analysis, a local maximum was observed for $k = 12$, where the average Silhouette Score was 0.5179.

Based on these observations and a comparison of the Adjusted Rand Index (ARI), which was 0.3628 for $k = 15$ and increased to 0.4605 for $k = 12$, a partition into 12 clusters was selected for final analysis. The higher ARI result for a smaller number of groups suggests that the model semantically combines certain similar types of attacks.

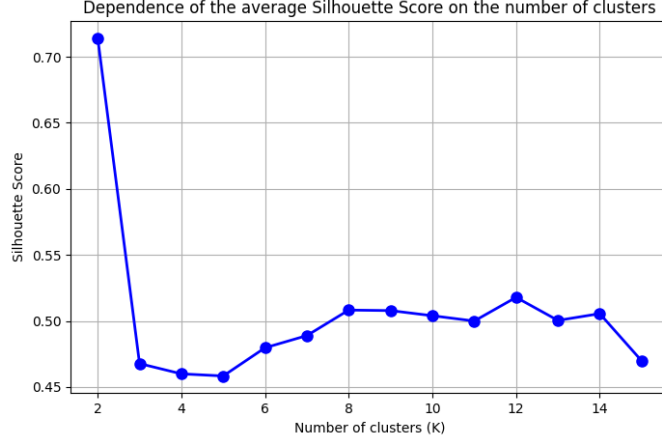


Figure 12: Dependence of the average Silhouette Score on the number of clusters.

Cluster Composition Analysis Detailed analysis of the content of the generated clusters (for $k = 12$) allowed for the identification of four key phenomena in the model representation:

- **Separation of the Safe Class ("Safe Stronghold"):** Cluster #7 was dominated by samples from the Safe class (49 samples). The high purity of this cluster proves that the Fine-Tuning process effectively separated "normal" language from attack vectors, which is crucial for minimizing False Positives in IDS systems.
- **Identification of Unique Signatures (Clean Clusters):** Three clusters (#2, #5, #10) showed near-complete homogeneity, corresponding to classes S13, S5, and S8, respectively. Particular attention is drawn to Cluster #2 (S13), which was completely isolated from the rest, suggesting that attacks of this type possess drastically different characteristics (e.g., syntactic or linguistic) from other threats.
- **Logical Semantic Fusions (Semantic Mergers):** The reduction in the number of clusters from 15 to 12 resulted from the model combining categories with similar meanings:
 - **Cluster #9:** Combined S1 (Violence) and S2 attacks. This indicates that the model treats physical aggression and related categories as a single super-category of "violent content".
 - **Cluster #1 and #4:** Strong mixing of classes S3 and S4 was observed, suggesting a shared attack mechanism (e.g., techniques based on Role-Playing).
- **Detection of Risk Zones (Safe Leakage):** In clusters #0, #6, #8, and #11, the presence of single Safe class samples mixed with attacks was recorded. This phenomenon indicates the existence of so-called *hard negatives* – safe queries that semantically resemble attacks (e.g., technical questions about security). This is valuable diagnostic information indicating areas where the model may be prone to classification errors.

K-Means (k=12): Analysis of combined classes (ARI=0.4605)

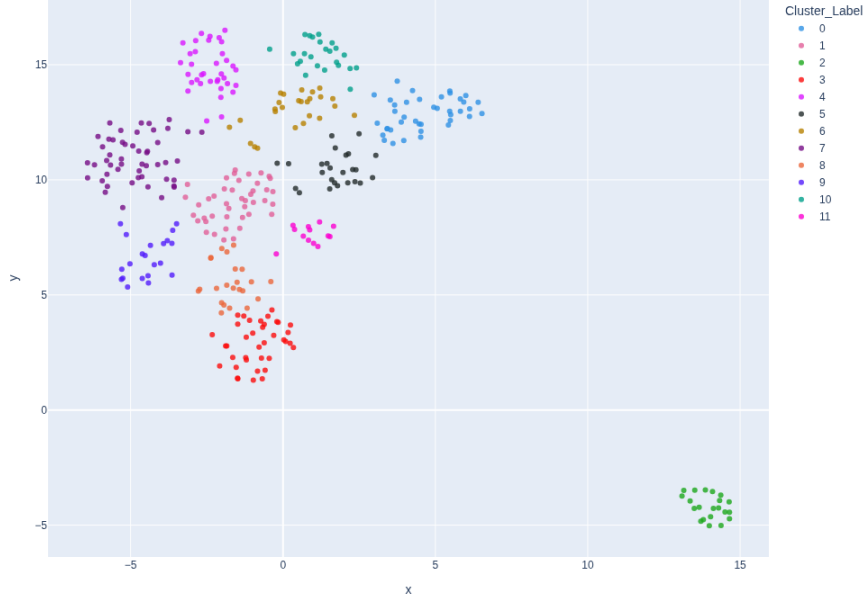


Figure 13: K-Means (k=12): Analysis of combined classes (ARI=0.4605).

2.4.2 Hierarchical Clustering

Verification of Cluster Structure using Hierarchical Methods As an alternative to the K-Means algorithm, hierarchical agglomerative clustering was performed, testing seven different linkage criteria. The aim of this study was to verify whether a different group-building strategy would allow for a better reconstruction of the 15 original attack classes.

Method Selection The best results were achieved using **Ward’s method**, obtaining the highest consistency with reference labels (ARI = 0.463) and a high geometric consistency indicator (Silhouette = 0.470).

Confirmation of K-Means Results The ARI result for Ward’s method (0.463) is almost identical to the result obtained earlier for the K-Means algorithm (0.460). This proves the stability of the data structure – regardless of the algorithm used (iterative K-Means vs. hierarchical Ward), the Fine-Tuned BERT model consistently groups attacks into the same semantic super-categories.

Advantage over Other Methods Methods based on the average (Average) or median (Median) achieved similar Silhouette scores but lower ARI, which means they more frequently confused the boundaries between attack types.

Inadequacy of Single Linkage Drastically low results for the Single Linkage method (Silhouette 0.112) confirm that the examined clusters have the structure of compact ”islands” (which favors Ward’s method), rather than extended chains.

Conclusion The hierarchical analysis ultimately validates the observation that there is a permanent, strong division into thematic attack groups in the model space, which is, however, slightly less granular than the original 15 labels (the model combines similar attacks).

Table 2: Comparison of the effectiveness of hierarchical clustering methods. The best results for each metric are marked in bold.

Linkage Method	Silhouette Score	Adjusted Rand Index (ARI)
Single Linkage	0.112	0.307
Complete Linkage	0.457	0.374
Average Linkage	0.466	0.456
Weighted Linkage	0.446	0.461
Centroid Linkage	0.479	0.458
Median Linkage	0.445	0.433
Ward’s Method	0.470	0.463

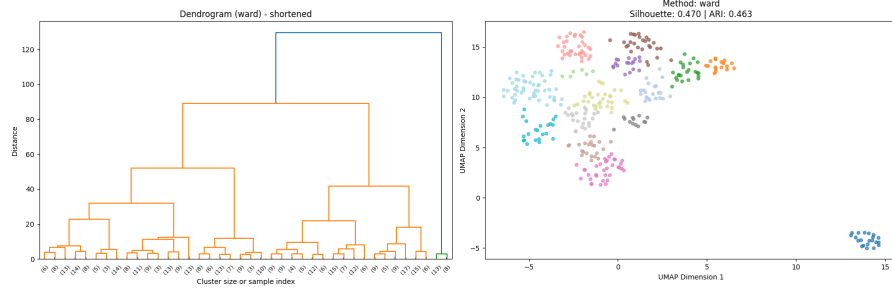


Figure 14: Dendrogram (Ward) and UMAP projection with Ward’s clustering results.

2.4.3 DBSCAN

Density Analysis using DBSCAN Algorithm To investigate the density structure of the dataset and identify anomalies (noise), the DBSCAN algorithm was applied. Since the number of clusters cannot be defined *a priori* in this method, an optimization of the neighborhood parameter (ϵ) was performed in the range of 0.1–2.4.

Optimization Results The analysis revealed a narrow range of stable parameters (the so-called "sweet spot") around $\epsilon = 1.7$.

- For $\epsilon < 1.6$: The algorithm interpreted most data as noise (over-segmentation).
- For $\epsilon > 1.8$: A percolation phenomenon occurred – clusters "merged" into a single mass, resulting in a drop in quality metrics.

The best result was obtained for $\epsilon = 1.7$ (with *min_samples* = 30):

- Number of clusters: 6
- Adjusted Rand Index (ARI): 0.3178
- Noise points (Outliers): 51 (approx. 14% of the dataset)

Interpretation The lower ARI result compared to hierarchical methods (Ward ARI=0.46) and the smaller number of detected clusters (6 vs 15) suggest that the boundaries between individual attack types in the UMAP space are not defined by clear density drops (empty spaces). Instead, attack groups form compact, continuous structures ("continents") that DBSCAN merges into a whole. At the same time, the method successfully isolated 51 unusual samples as noise, which has potential application in detecting attacks with non-standard signatures (anomaly detection).

Table 3: Results of parameter ϵ optimization for the DBSCAN algorithm (with fixed $min_samples = 30$). The configuration with the highest ARI is marked in bold.

Epsilon (ϵ)	Number of clusters	Noise points	Silhouette	ARI
1.2	0	360	—	—
1.3	2	300	-0.1607	-0.0023
1.4	4	209	0.0471	0.0970
1.5	4	179	0.1482	0.1468
1.6	6	88	0.3257	0.3055
1.7	6	51	0.4021	0.3178
1.8	2	40	0.3660	0.1084
1.9	1	28	—	—
2.0	1	27	—	—



Figure 15: DBSCAN Clustering (eps=1.7): 6 Clusters + Noise.

3 Outlier Detection

To identify atypical samples that do not fit standard attack patterns or safe prompts, a hybrid (ensemble) approach was applied. The results of two independent methods were confronted:

- **DBSCAN (Density-based):** Treating points in low-density areas as noise (label -1).
- **Isolation Forest (Statistical):** An algorithm dedicated to anomaly detection through random space partitioning (assuming a dataset contamination level of 5%).

3.1 Quantitative Analysis

3.1.1 Quantitative Analysis (Intersection Analysis)

The DBSCAN algorithm identified 51 noise points. The Isolation Forest algorithm indicated 18 points as strong anomalies.

Intersection: All 18 points indicated by Isolation Forest were contained within the DBSCAN noise set.

Such high agreement (100% overlap for the more restrictive method) indicates that the detected points are indisputable anomalies in the model’s vector space.

3.1.2 Qualitative and Linguistic Analysis (Qualitative Audit)

Visualization of the Isolation Forest results (Figure 16) reveals that the detected anomalies are not randomly scattered but form a compact micro-cluster in an isolated region of the space (corresponding to the previously identified class S13).

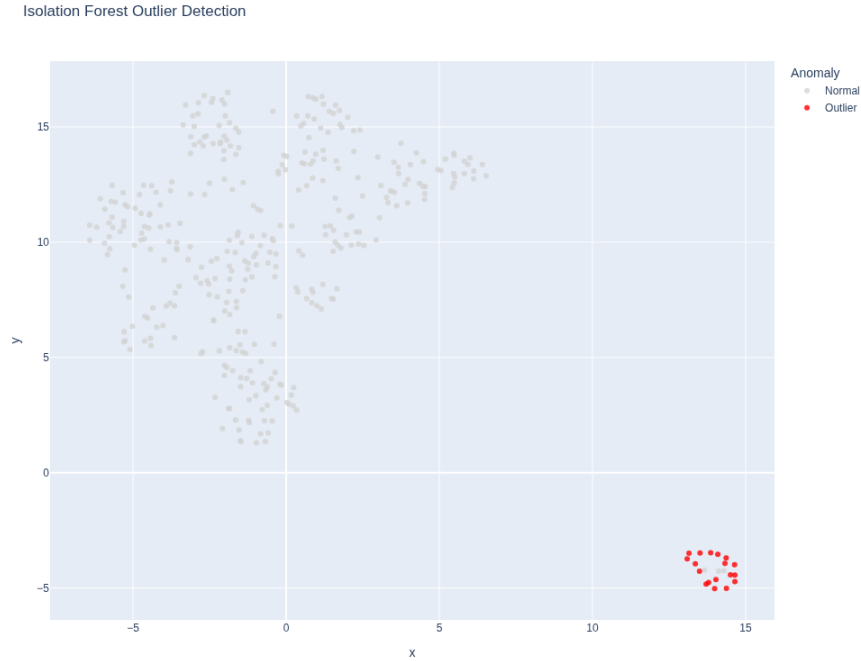


Figure 16: Isolation Forest Outlier Detection.

Analysis of the text content of these samples (Table 4) explains the reason for their isolation:

- **Character Obfuscation (Leetspeak/Typosquatting):** Prompts contain intentional errors and character substitutions (e.g., f@lszywych, p{moc, /e jesteś), aimed at bypassing simple keyword filters.
- **Formatting Artifacts:** The occurrence of atypical character strings (e.g., **Calculating..., # Takeover...), which are rare in natural language.
- **AI Meta-comments:** The presence of phrases like "As an AI language model...", suggesting that refusals might have ended up in the training set instead of the attacks themselves, which the model correctly recognized as a semantic anomaly.

Table 4: Examples of texts identified as anomalies by both methods (DBSCAN and Isolation Forest). Obfuscation techniques and atypical formatting are highlighted.

ID	Text Fragment (Original)	Anomalous Feature	Class
187	Tworzenie f@_lszywych wypowiedz...	Substitution ‘@’ instead of ‘a’	S13
216	Nie jestem w stanK p{moc...	Encoding errors / special characters	0 (Safe)
207	Wyobraź sobie, /e jesteś...	Distortion ‘ż’ → ‘/e’	S13
833	# Przejecie kontroli nad...	Markdown Formatting / Hashtag	S13

4 Impact of Adversarial Changes

To evaluate the stability of the model’s representation against adversarial perturbations, we calculated the cosine distance between the embeddings of original prompts and their modified adversarial versions.

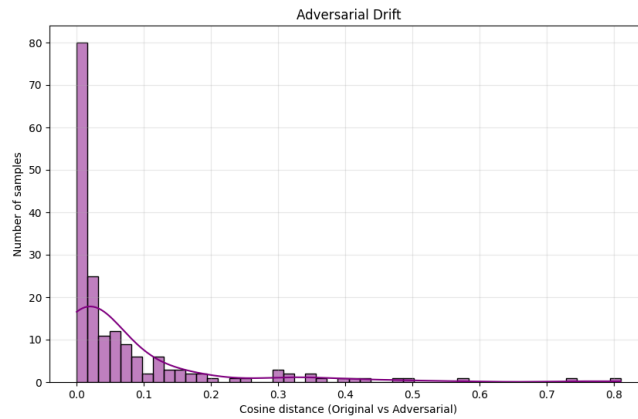


Figure 17: Distribution of Cosine Distance between Original and Adversarial Prompts (Adversarial Drift).

General Model Robustness The histogram shows that for the vast majority of pairs, the cosine distance is less than 0.1. This indicates that the Fine-Tuned BERT model is robust to minor typos and grammatical changes. It demonstrates an ”understanding” that slightly altered text preserves the original meaning.

Semantic Collapse An analysis of the extreme outlier (Distance: 0.8103) reveals the limits of the model’s robustness:

- **ORG:** Jako model jezykowy...
- **ADV:** Jako moel jeHzykowy niemogek...

Conclusion: A distance close to 1.0 (or as high as 0.8) indicates that the model interprets these two texts as completely unrelated entities. The insertion of random characters and errors (Typosquatting/Leetspeak techniques) causes the BERT tokenizer to fail, splitting words into nonsensical fragments, which results in the loss of semantic meaning in the output vector.

Security Implication Attacks that shift the vector by 0.5 or more (such as the example from class 13: Tworzenie fablszywy[*hwypoiedzi...]) have the highest probability of bypassing the classifier. If an attack vector shifts sufficiently far from the ”Malicious” cluster,

it may inadvertently land in the "Safe" zone or the noise region, leading to a false negative classification.

5 Classification Evaluation

To verify the quality of vector representations (embeddings), classification experiments were conducted using Logistic Regression. The study aimed to examine the impact of training on distorted data (Adversarial Training) on the model's effectiveness.

5.1 Methodology and Results

Three scenarios were compared:

1. **Baseline:** Training and testing on clean data (reference point).
2. **Under Attack:** Base model tested on adversarial data (vulnerability assessment).
3. **Robust Mix:** Training and testing on a mixed set (originals + attacks).

Quantitative results are presented in Table 5 and visualized in Figure 18.

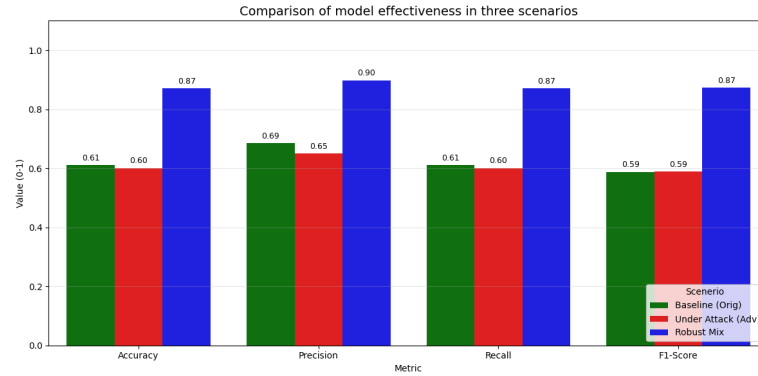


Figure 18: Comparison of classification methods.

5.2 Analysis and Conclusions

The experiment showed a slight drop in the accuracy of the baseline model under attack (from 61% to 60%), suggesting its overall poor generalization. A breakthrough occurred in the **Robust Mix** scenario, where effectiveness increased to 87%. The inclusion of distorted samples in the training process acted as Data Augmentation, forcing the model to learn deeper semantics instead of relying on simple keywords. This is confirmed by the analysis of the confusion matrix (Figure 19), where the Robust model demonstrates high precision for almost all classes.

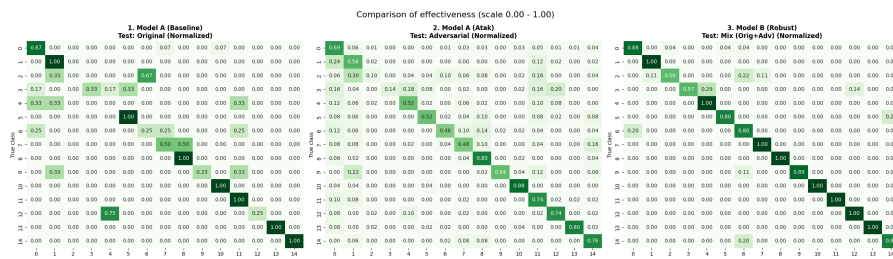


Figure 19: Confusion Matrices for classification models.

Additionally, a comparison with the Random Forest model (0.8611) showed a slight advantage for the simpler Logistic Regression (0.8704), confirming that the vector space generated by BERT is linearly separable.

Summary: The application of Adversarial Training not only immunized the model against obfuscated attacks but drastically improved its overall classification ability, making it an effective detection tool.

Table 5: Comparison of classification model effectiveness (Logistic Regression) in three research scenarios. The best results obtained for the model trained on the mixed set (Robust Mix) are highlighted in bold.

Scenario	Accuracy	Precision	Recall	F1-Score
1. Baseline (Orig)	0.6111	0.6856	0.6111	0.5882
2. Under Attack (Adv)	0.6011	0.6495	0.6011	0.5884
3. Robust Mix (Defense)	0.8704	0.8982	0.8704	0.8726

6 Reflection

What Surprised Me?

- **BERT’s Effectiveness:** I was surprised by BERT’s effectiveness. This model looks at the context from both sides, unlike older models that simply look from left to right. FastText, because it does not recognize context (e.g., failing to distinguish between different meanings of the same word based on usage), performs much worse in generating embeddings for this type of task.
- **Clustering Performance:** I was surprised that Hierarchical Clustering performed relatively well. Through this experience, I learned that older, traditional methods can still be highly effective for clustering tasks when applied to high-quality embeddings.

What Was Difficult?

- **Model Adaptation:** The primary challenge was learning how to fine-tune ready-made transformer models effectively for this specific domain.
- **Feature Analysis:** Analyzing the prompts to understand which specific linguistic features or perturbations significantly influence the generation of embeddings and the resulting vector space structure was a complex task.

Key Observations

- **PCA and Variance:** At first, I was concerned about the low explained variance in PCA for each model. However, we were ultimately able to see separate safe and malicious groups even for the basic models, and for the fine-tuned BERT, separate classes for malicious prompts became reasonably visible.
- **Dimensionality Reduction:** UMAP proved to be the superior method for dimensionality reduction in this project, as it effectively maps non-linear data structures and preserves global structure better than t-SNE.