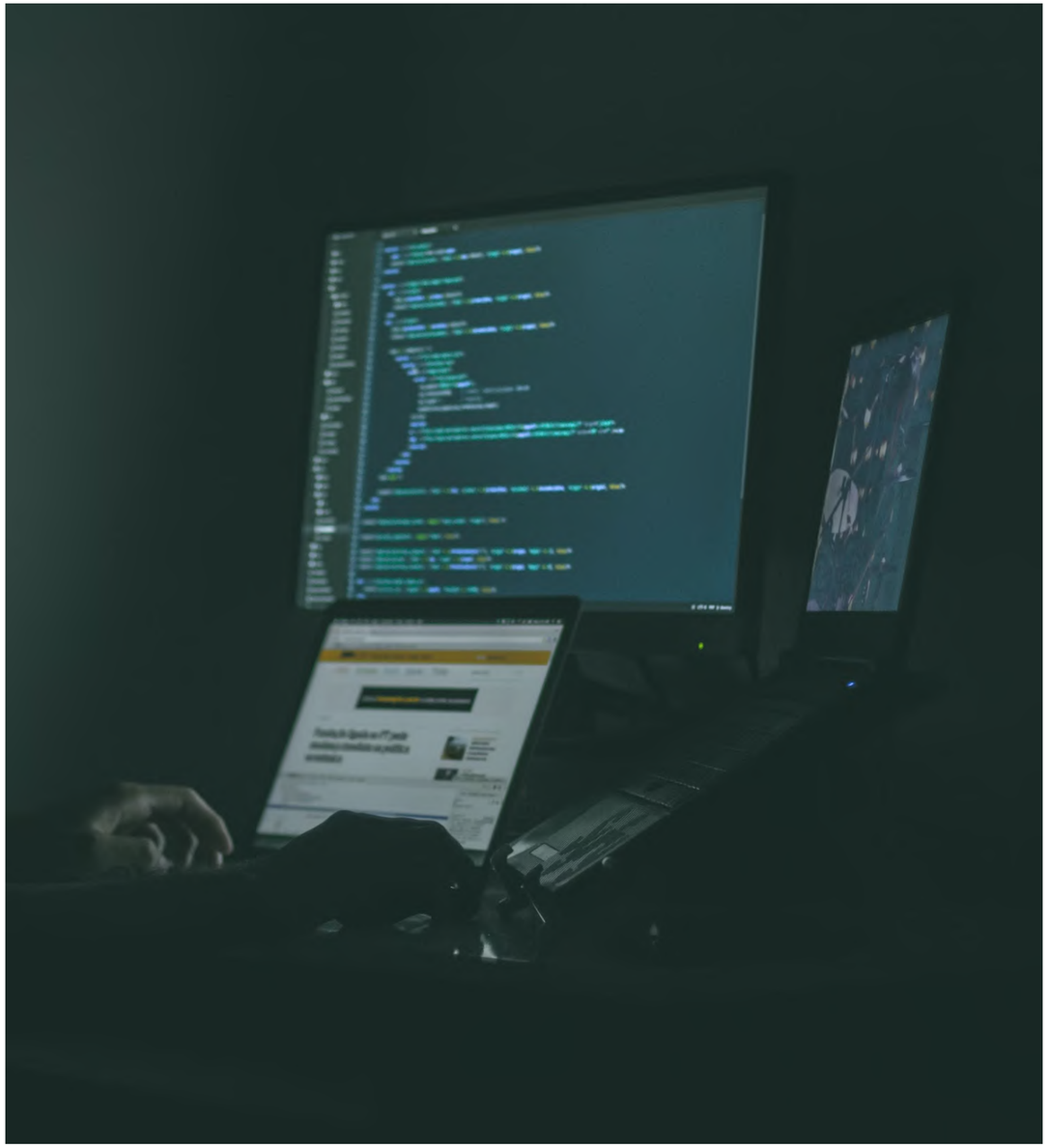


Classification of harmful content

Text analysis and classification based on natural language processing methods

Agenda

1 Introduction

2 Text analysis

3 Classification of harmful content

4 Results and conclusions

Introduction

The section focuses on defining the problem addressed and describing the potential solution.
The methods and techniques used and the available dataset will be briefly presented.

Description of the issues addressed

01

Problem

Cyberbullying, phishing, harmful and hurtful content are a growing and disturbing trend in social media and online communications. This phenomenon poses serious challenges as customers are at risk of losing data and financial resources.

02

Solution

Detecting and responding to these threats by classifying harmful content is a potential solution to protect customers. For text classification tasks, natural language processing (NLP) combined with machine learning (ML) plays a key role.

03

Methods and tools

In order to create an effective tool, typical NLP techniques such as n-grams and TF-IDF, and machine learning models such as SVM, NB and HerBERT were used. The implementation of these solutions was carried out in the freely available Python programming language.

04

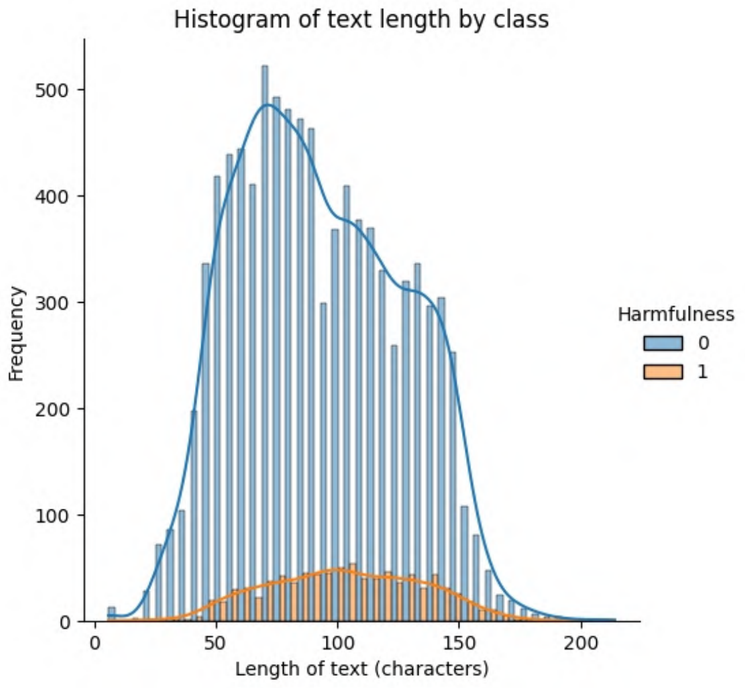
Data

The tool was created based on a common dataset from the evaluation campaign for NLP tools PolEval 2019. The dataset includes neutral and harmful content written by users (tweets). These include cyberbullying, hate speech and related phenomena.

Text analysis

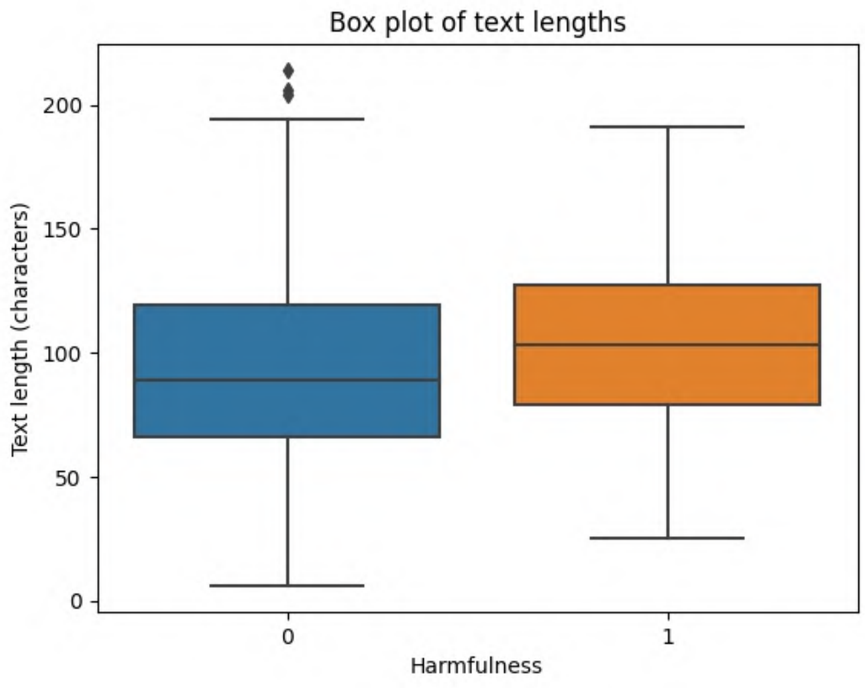
The section focuses on studying the content of tweets by examining the length of sequences, the occurrence of specific features in texts, and analysis of n-gram models.

Text length analysis



Differentiated distributions

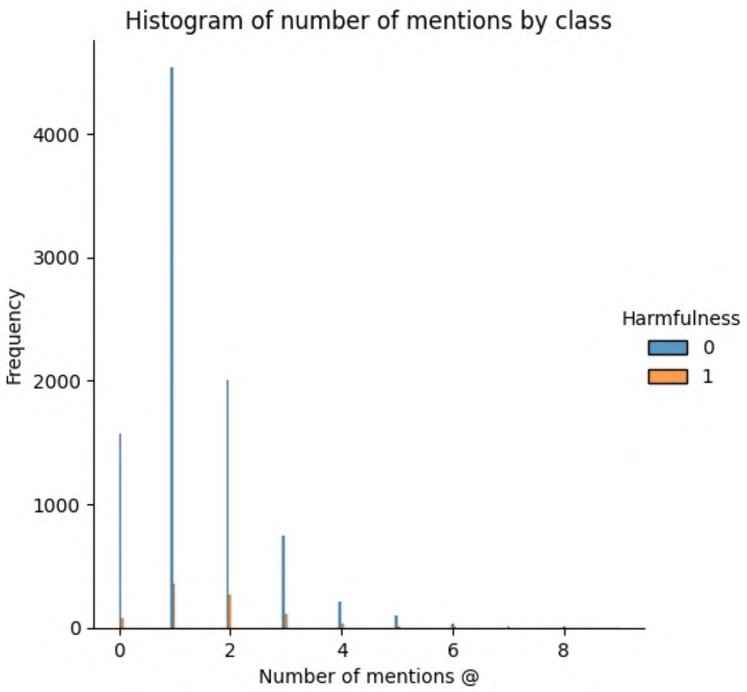
The distribution of harmful tweets is more flattened with less concentration. In contrast, harmless content has a concentrated slender distribution. Both are relatively symmetrical.



Outliers that occur

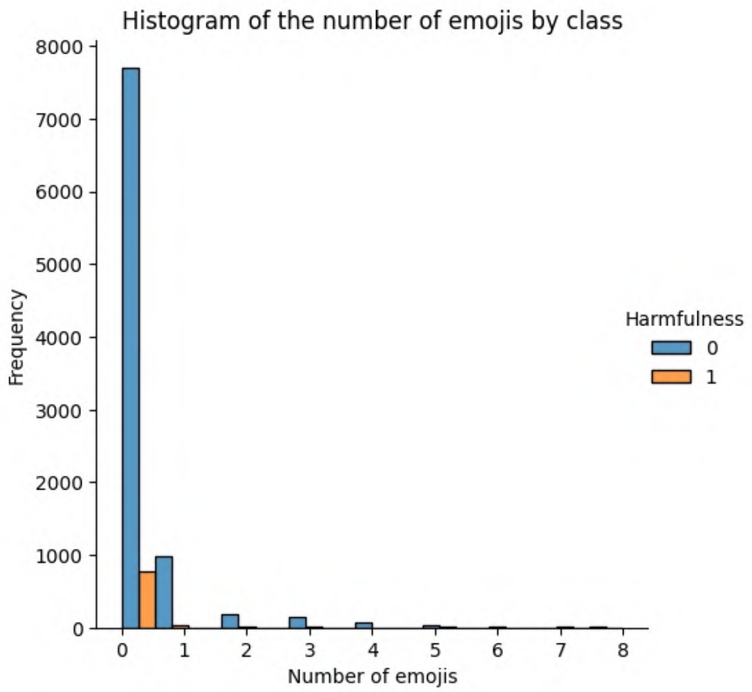
The average length of the harmless texts is smaller, and the whiskers are slightly longer compared to the harmful ones. In addition, outlier observations can be noted among them.

Analiza ukrytych cech w tekście



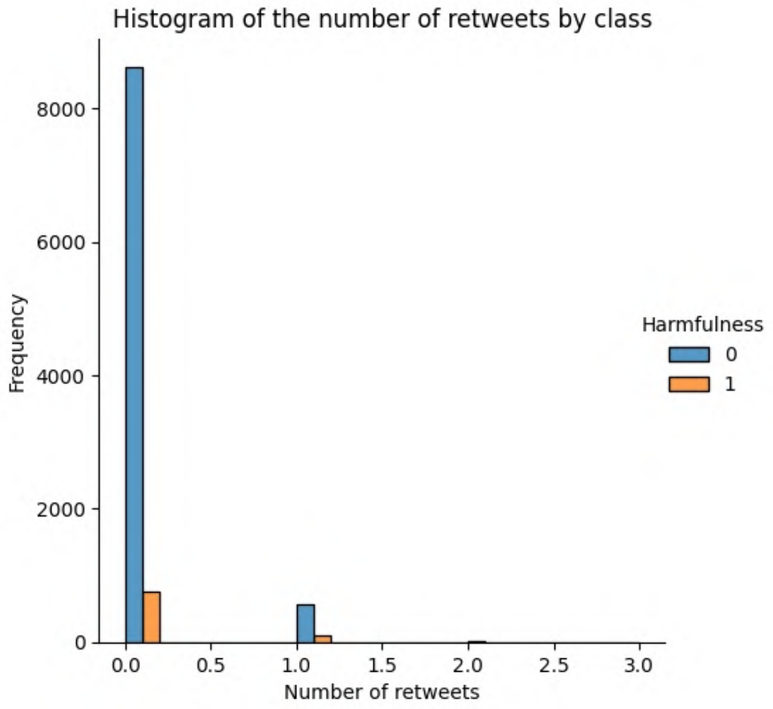
Number of markings

The graph of the frequency of mentions in tweets shows no specific property. The difference in frequency is due to unbalanced classes. There are also outlier observations.



Number of emoticons

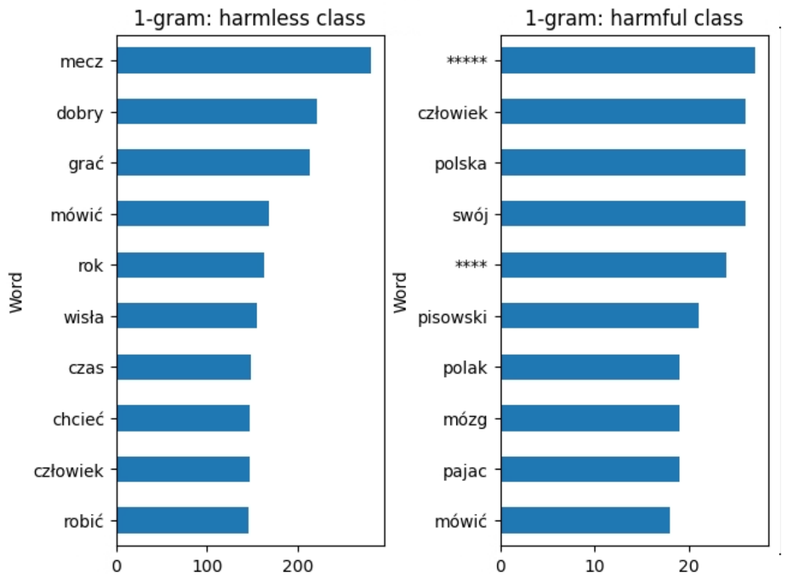
A graph of the frequency of emoji icons in texts reveals a potentially important feature. In non-harmful texts, emoticons occur more frequently and in greater quantities. In contrast, for harmful tweets, emoji emoticons are mostly absent.



Number of retweets

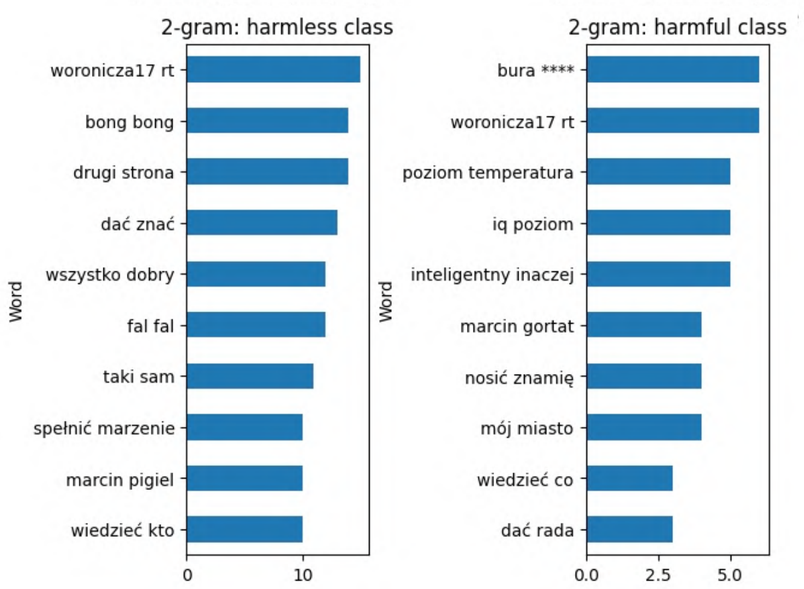
A histogram of the number of retweets by class does not imply specific conclusions. Both classes contain retweets with a natural majority proportion of harmless tweets resulting from the unbalanced dataset.

Analiza modeli n-gram



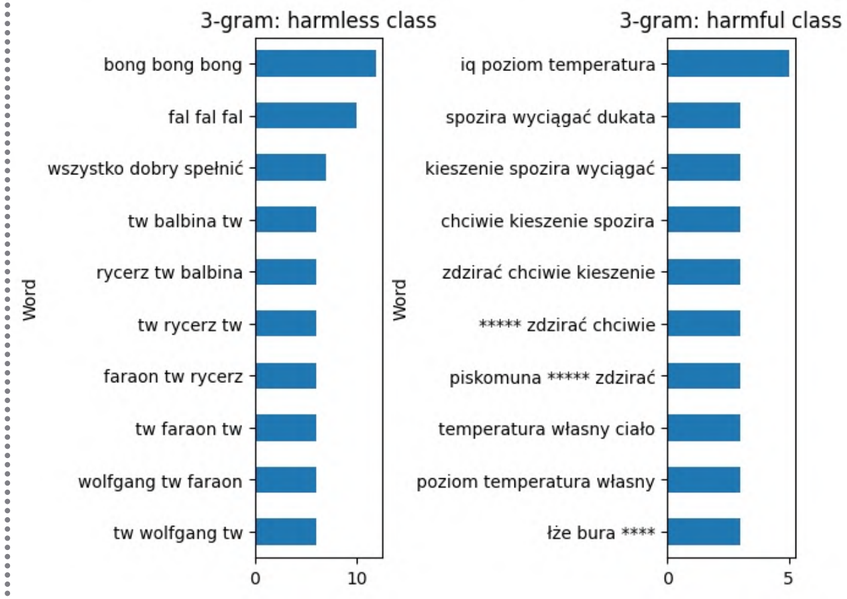
Uni-gram

In harmful texts, some of the most common words are vulgarisms. In addition, there are references to the nation and politics. There are common parts.



Bi-gram

One of the most common bigrams is a television program broadcast on public television. In harmful texts further down the line, vulgarity is the most common. There are also phrases referring to the level of intelligence.



Tri-gram

No common part. Harmful content mainly expresses vulgarisms and negative words like greed. There are also political references.

Classification of harmful content

The section focuses on the process of building a model to classify negative texts. Methods used include TF-IDF, EDA, SMOTE, classical Naive Bayes type models, SVMs and HerBERT-type deep learning models. Evaluation was done using classification metrics (ACC, Precision, Recall, F1) and the AUC ROC curve.

Vectorization and oversampling

Vectorization is the process of converting words or documents into numerical vectors. A TF-IDF method was used to calculate the importance of words in a given document based on their frequency of occurrence (TF) and the inverse frequency of their occurrence in all documents in the set (IDF).

Because of the unbalanced dataset (small number of harmful texts), oversampling was used, which equalizes the level of classes by including new observations from the minority class. Methods used were Easy Data Augmentation (EDA), which involves synthetically creating a new sample of a given class through simple transformations on the text, and SMOTE, which equalizes the level of classes in the set by drawing observations from the minority class along with its neighbor to form their convex combination.

```
# Set variables with features and label
X_tr = train.text
y_tr = train.label

# Initialize TF-IDF vectorizer
vectorizer = TfidfVectorizer()

# Fit and transform training features
X_tf = vectorizer.fit_transform(X_tr)

# Transform test features
X_test_res = vectorizer.transform(test.text)

# Initialize SMOTE technique at given seed
sm = SMOTE(random_state=42)

# Fit and transform training features and labels
X_res, y_res = sm.fit_resample(X_tf, y_tr)
```

```python
# Set parameters values
estimator = Pipeline([
        ('tfidf', TfidfVectorizer()),
        ('svm', SVC()),
    ])

# Specify grid with parameters
param_grid = {
    'tfidf__ngram_range': [(1, 1), (1, 2), (1, 3)],
    'tfidf__norm': ['l2'],
    'tfidf__smooth_idf': [True],
    'tfidf__sublinear_tf': [False],
    'svm__C': [0.1, 1, 10],
    'svm__kernel': ['linear', 'rbf'],
}

# Perform CV
svm_model, svm_results, grid_search = cm.perform_cross_validation(X, y, estimator, param_grid)

cv_train_results, cv_val_results = {}, {}
for k in svm_results.keys():
    metric = k.split('_')[-1]
    if 'mean_train_' in k:
        cv_train_results[metric] = np.mean(svm_results[k])
    if 'mean_test_' in k:
        cv_val_results[metric] = np.mean(svm_results[k])

print("Train CV results:\n", cv_train_results)
print("Validation CV results:\n", cv_val_results)

# Get predictions
y_pred = svm_model.predict(test.text)
y_test = test.label

# Evaluate results
scores_svm = cm.score(y_test, y_pred)
print("Test dataset results:\n", scores_svm)

# Save model
dump(svm_model, 'models/svm_model.joblib')
```

Classical models

Two underlying machine learning models for the classification task were trained: a Naive Bayes classifier (NB) and a support vector machine (SVM). In doing so, different combinations relative to oversampling were combined. Training was done with tuning of the model's hyperparameters using 3-stage cross-validation. On the left, a visible code snippet for EDA-based SVM model training.

Naive Bayes classifier is a simple and efficient classification algorithm that is based on the application of Bayes' rule with the assumption of independence of features. It assigns a new observation to the appropriate class based on the probability of its features in each class.

Support vector machine (SVM) is a machine learning algorithm that is used for classification, among other things. It determines the optimal hyperplane in a multidimensional space to best separate data belonging to different classes.

Deep learning models

The HerBERT deep learning model was also trained. This is a BERT-type model, which is a bidirectional encoder representation of the transformer. Models based on transformer architecture in recent times are SOTA-type models in the NLP domain. HerBERT is a pre-trained model on the Polish text corpus. The output of the model is the encoded content of documents in vector form called embedding. With training, it preserves the meaning concept of words unlike basic vectorization techniques. It can be used for semantic search.

The model was tuned on available data of harmful and neutral content. A sigmoidal function preceded by a dropout layer was applied to the output layer for regularization. To the right is a code snippet from training the model, along with the assumed values for learning rate, batch size and number of epochs.

```
# Set number of epochs, batch size and learning rate
EPOCHS = 5
BATCH_SIZE = 4
LR = 2e-05

# Initialize HerBERT model
model = cm.HerBertForSequenceClassification(num_classes=1, dropout_rate=0.5)
model_custom_name = 'herbert_model'

# Start model training
results, model = cm.train_model(
    model = model,
    train_data = df_train,
    val_data = df_val,
    learning_rate = LR,
    epochs = EPOCHS,
    batch_size= BATCH_SIZE,
    custom_model_name=model_custom_name
    )
results['model'] = model_custom_name

# Evaluate model on test dataset
test_res = cm.evaluate(
    model = model,
    test_data = df_test
    )
test_res['model'] = model_custom_name

# Save model
checkpoint = {
'model': cm.HerBertForSequenceClassification(num_classes=1, dropout_rate=0.5),
'state_dict': model.state_dict()
}
torch.save(checkpoint, 'models/' + model_custom_name + '.pth')
```

Results and conclusions

The section focuses on presenting the results obtained results and lessons learned, including business lessons.

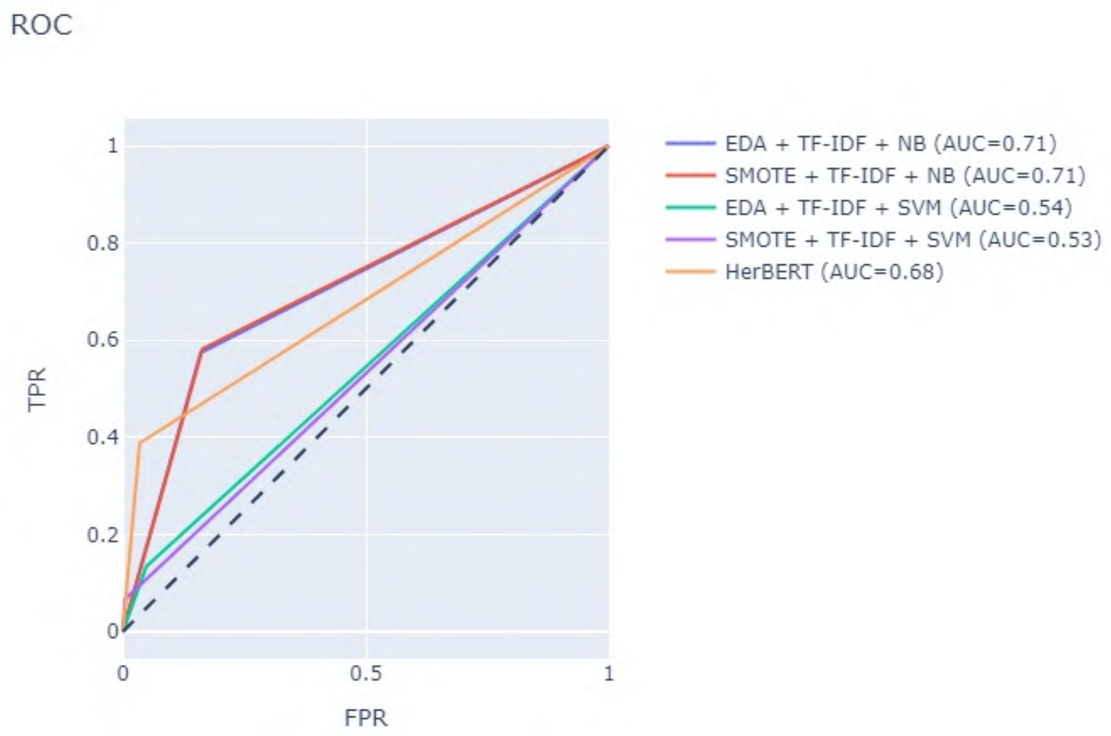
Results and recommendation

The best results were achieved by NB models in both oversampling combinations and HerBERT. In the case of NB, it is preferable to choose the EDA technique due to the simplicity and interpretability of synthetically generated new samples.

HerBERT achieved the highest F1 score value (which is the main evaluation metric in the PolEval 2019 competency). Compared to the NB model, it has a higher precision value, but a lower recall value. This means that NB finds harmful content more effectively at the expense of precision, while HerBERT detects this content more accurately, but in smaller amounts. The NB model also achieved a higher AUC score.

Based on the assumption that it is more valuable to detect harmful content at the expense of precision, the NB model is recommended. Moreover, the model's results are easy to justify, as it is based on one of the basic theorems of probability calculus.

Model	ACC	Precision	Recall	F1
EDA + TF-IDF + NB	**0.793**	**0.339**	**0.575**	**0.427**
SMOTE + TF-IDF + NB	0.791	0.338	0.582	0.427
EDA + TF-IDF + SVM	0.842	0.3	0.134	0.186
SMOTE + TF-IDF + SVM	0.871	0.692	0.067	0.122
HerBERT	**0.888**	**0.634**	**0.388**	**0.481**



Conclusions

SVM needs high-quality synthetic samples

SVM is known for its high efficiency with multivariate data and for working on small samples as in this case (due to a kernel trick). Nevertheless, it performed worst in the experiment, most likely due to oversampling. The dataset required the production of a large number of artificial observations which introduced noise in the multidimensional feature space and over-fitting the model to the training data (over-fitting).

HerBERT can generalize on a small sample

HerBERT is a model pre-trained on a Polish corpus of text data. It only requires tuning on the new data under consideration, and the experiment shows that a huge amount of data is not required. Moreover, they do not have to be of the best quality (as for SVMs), since in the case of DL models, features from the data are extracted at the learning stage. However, this situation occurs when the new data is of a similar nature to the data on which it was trained.

NB classifier does not depend on oversampling method

The Naive Bayes classifier achieved the same results for both oversampling methods, which leads us to conclude that they do not significantly affect the results obtained with this model. In both techniques, the input feature vector space was completed with relatively similar synthetic samples (SMOTE creates combinations, and EDA consists of transformed initial data) which may explain this situation. Nevertheless, an experiment on a wider range of data balancing techniques would need to be conducted to confirm.

Business applications

01

Effective customer security

The use of NLP models combined with machine learning, such as Naïve Bayes and HerBERT, enables institutions to effectively detect malicious content. The experiment carried out can be extended with new data to detect cyber-bullying and phishing, allowing quick response to threats and effective protection of users against loss of data and funds.

02

Minimize reputational risk

Classifying harmful content using advanced NLP techniques and ML models allows institutions to monitor and identify potential threats to their reputation. Earlier detection and effective response to negative content (for example, on social media) will minimize the risk of image damage.

03

Improving user interaction

The use of classification models allows analysis of user-generated content, which can provide valuable information about user preferences and needs. Institutions can use this data to personalize services, increase user engagement and improve service quality, which can help strengthen relationships with users and build brand loyalty.

References

1. Code available in the repository: https://github.com/szpwski/nlp_poleval_text_classification
2. PolEval 2019 dataset from text classification task 6. http://2019.poleval.pl/index.php/tasks/task6
3. Ogrodniczuk M, Kobyliński Ł.: *Proceedings of the PolEval 2019 Workshop*, Instytut Podstaw Informatyki PAN (2019).
4. Padurariu C, Breaban M. E.: *Dealing with Data Imbalance in Text Classification*, Procedia Computer Science (2019).
5. Wei J, Zou K.: *EDA: Easy Data Augmentation Techniques for Boosting Performance on Text Classification Tasks*, Association for Computational Linguistics (2019).
6. Murphy K.: *Probabilistic Machine Learning: An Introduction*, The MIT Press (2022).
7. Mroczkowski R, Rybak P, Wróblewska A, Gawlik I.: *HerBERT: Efficiently Pretrained Transformer-based Language Model for Polish*, Association for Computational Linguistics (2021).