Intelligent Systems
Seminar Assignment 2 – Detecting Phishing Emails

1 Introduction

In this seminar assignment, your goal is to build natural language processing (NLP) systems that detect phishing emails: messages that try to trick people into revealing passwords, opening malicious links or sending money. You will apply the methods from the NLP lab, moving from classical machine learning on TF-IDF and word embeddings to transformer models that you use off the shelf and then fine-tune, and write a report that compares these approaches in a clear and concise manner.

You can do the assignment in groups of 2 or individually. You are allowed to use artificial intelligence or agentic systems for help. If you do, you are required to submit a log of your dialogue with the system. You will also be required to justify and explain your choices during the seminar assignment presentations that will take place during the labs the week following the submission deadline.

2 Data

You will use the Phishing Email Dataset on Hugging Face (a copy of the Kaggle dataset *Phishing Email Detection*), shared under the LGPL-3.0 licence. It contains about 18,700 English emails, each labelled as a safe email or a phishing email. Load it with `load_dataset('zefang-liu/phishing-email-dataset')`, which gives a single `train` split with the columns `Email Text` and `Email Type` (values `Safe Email` and `Phishing Email`). If `load_dataset` fails, read the CSV directly with `pd.read_csv('https://huggingface.co/datasets/zefang-liu/phishing-email-dataset/resolve/main/Phishing_Email.csv')`.

Real security data is messy. Before modelling, check for empty texts, duplicates and leftover HTML; handling them is part of Task 1.

3 Task 1 – Exploration, cleaning and anonymisation (15 %)

In the first task, you will explore the data and prepare the texts for modelling, as in the NLP lab.

- **Exploration.** Report how many emails each class has and how long they are. Count missing and duplicated texts and remove them before splitting: a duplicate that lands in both the training and the test set inflates your results. Compare a simple security signal between the classes, such as the number of links per email, and visualise your findings.
- **Cleaning.** Write a preprocessing function based on the lab: lowercase the text, remove HTML tags and punctuation with regular expressions, tokenise it, remove stopwords and lemmatise with NLTK. Links are a strong phishing signal, so make two versions: one that removes URLs and one that replaces each URL with a token such as `URL`.
- **Anonymisation.** Emails contain personal data. As in the lab exercise, use Stanza's named-entity recognition to replace person names with a placeholder such as `[PERSON]` in a random sample of 200 emails, and show a few examples. Explain why anonymisation matters when security data is shared.

- **Splits.** Split the data into stratified training, validation and test sets (for example 70/15/15) with a fixed random seed. Use the same test set in every task.

4 Task 2 – Classical machine learning: TF-IDF and word embeddings (25 %)

In the second task, you will build classical machine-learning baselines from the text representations in the lab. These TF-IDF and word-embedding features are simple and fast; the transformer models in Task 3 are compared against them. Experiment with several classifiers (for example logistic regression, a linear SVM, naive Bayes, a random forest and gradient boosting) and find the one that works best.

- **TF-IDF.** Train your classifiers on TF-IDF features, and compare the two cleaned versions of the text from Task 1: URLs removed or replaced with a token.
- **Your own word embeddings.** Train a Word2Vec model with gensim on the training emails. Represent each email as the average of its word vectors and classify it. Compare at least two settings, for example CBOW versus skip-gram, or two vector sizes.
- **Pretrained word embeddings.** Repeat this with pretrained English vectors from gensim's downloader, as in the lab exercise, for example `glove-wiki-gigaword-100` or `word2vec-google-news-300`.
- **Inspect the embeddings.** For words such as "password", "account" and "verify", show the most similar words in your own and in the pretrained embeddings. Plot a 2D PCA projection of selected words, as in the lab.
- **Tuning.** Tune several hyperparameters of your best classifiers with cross-validation on the training set, and describe which settings helped.

5 Task 3 – Transformer models (30 %)

In the third task, you will use transformer models, first off the shelf and then fine-tuned on the data. This is the core of the assignment.

- **Zero-shot baseline.** As a quick baseline, classify a fixed random subset of the test set (for example 500 emails, truncated as in the lab) with a pretrained `pipeline("zero-shot-classification")` model such as facebook/bart-large-mnli and candidate labels such as "phishing email" and "legitimate email". Try at least two sets of candidate labels and report how much the results change.
- **Fine-tune a transformer.** Fine-tune a pretrained model such as `distilbert-base-uncased` (or `bert-base-uncased`) on your training set with the Hugging Face `Trainer`. These models read at most 512 tokens, so truncate longer emails. Use the validation set to choose the learning rate and number of epochs, and plot the training and validation loss. This is the main part of the task.
- **Compare.** Evaluate the fine-tuned model on the same test set and metrics as the earlier tasks, and compare it with the zero-shot baseline and with your best classical model from Task 2. Does fine-tuning help, and by how much?
- **Ready-made detectors and leakage.** Several models on Hugging Face were already fine-tuned on this very dataset. Explain why testing such a model on your test set would give misleadingly good results, and how you could check whether a model has already seen your test data. You do not need to run such a model.

Fine-tuning needs a GPU; use a free notebook service such as Google Colab or Kaggle. If training is slow, you may fine-tune on a subset of the training data; report how large it is.

6 Task 4 – Evaluation and report (20 %)

In this task, you will evaluate your approaches and create a report (in the form of a Jupyter notebook) that contains your code and the results. To perform a comprehensive evaluation, you should:

- **Compare all approaches.** Build one table with accuracy, precision, recall and F1 for the phishing class, and training time, for your best classical model from Task 2 and for the transformer models (zero-shot and fine-tuned) from Task 3. Show it as a bar chart. For the classical models, also report the mean and standard deviation of the cross-validation scores.
- **Think like a security team.** Report how many safe emails are flagged as phishing (false positives) and how many phishing emails are missed (false negatives). Discuss which mistake costs an organisation more, and show how both change when you move the decision threshold of your best model.
- **Analyse the errors.** Show the confusion matrix of your best model. Read about ten phishing emails it missed and describe what made them hard to detect.
- **Explain a model.** For a TF-IDF logistic regression model, show the words with the largest weights for each class. Do they match known phishing cues, such as urgency or requests to verify an account?

You will present your report during the seminar presentations that will take place during the labs the week after the submission deadline. Your report should include:

- The code of your approach separated by task. Highlight some key segments of the code that you think are especially interesting or important to your approach.
- The dialogue with an agentic system or AI you used to build the solution, obtain the results, and/or write the report.
- The results of each task.
- Comparisons of the different approaches.

Make sure to present these results in a clean and concise manner, using tables and graphs that are easy to read and understand. The presentations will last 8 to 10 minutes, so make sure your report is thorough but concise.

7 Task 5 – Out of the box: the devil's approach (10 %)

Put yourself in the attacker's shoes: you know exactly which model an organisation uses, because you built it in the tasks above. Can you write a phishing email that still reads as a convincing phishing email to a person, but that your own best classifier labels as safe?

This is white-box evasion, a standard idea in adversarial machine learning: you have full access to the model and use it to find its blind spots. The goal is to understand and then close those blind spots, not to produce a usable attack.

- **Rules of engagement.** Work only against your own model, on your own machine, with the dataset's emails. Do not send anything to anyone and do not use a real brand, person or working link. A phishing email here is a text file you analyse, never a message you deliver.
- **Craft evasions.** Start from phishing emails in the test set that your model catches, and change them so the model flips to safe. Try at least two strategies, for example: add filler text that looks safe (greetings, a signature, newsletter-style sentences), swap the words with the largest phishing weights for synonyms or look-alikes, or move the payload into a form the bag-of-words model cannot read well. For each attempt, keep the email recognisably phishing to a human reader.

- **Measure the gap.** On a set of such crafted emails, report how far your model's phishing score drops and how many flip to safe. Do the same emails also fool the other models from Tasks 2 and 3? A gap that one model misses but another catches is the interesting result.
- **Close the door.** Propose and test at least one defence. Examples: add some of your crafted emails to the training set and retrain (adversarial training), normalise look-alike characters before classifying, or add a feature the attack cannot easily hide, such as the presence of a link or a request for credentials. Report whether the defence holds against a fresh round of evasions.
- **Reflect.** In a few sentences, discuss what this says about deploying such a filter in a real organisation, and about the limits of text-only phishing detection.

Marks reward a clear attack-and-defence story with evidence, not the most damaging email. Keep the tone analytical; this is a red-team exercise against your own system.

8 Submission

- **Deadline:** to be announced on učilnica.
- **Format:** a Jupyter notebook with your code and report.
- **Group work:** groups of 2, or individually.
- **Use of AI:** allowed. Submit the full log of your dialogue with the AI or agentic system as a single .txt file together with your notebook.
- **Presentation:** 8 to 10 minutes, during the labs the week after the deadline. Be ready to justify and explain your choices.