Technická univerzita v Košiciach

Fakulta elektrotechniky a informatiky

Katedra kybernetiky a umelej inteligencie

Optimalizácia písania diplomových prác na našej fakulte

Diplomová práca

Príloha C

Používateľská príručka

Vedúci diplomovej práce:
prof. Ing. Kristína Machová, Ph.D.

Diplomant:
Patrik Gecík

Košice 2025

Obsah

1 Funkcia programu

Program vyvinutý v rámci tejto diplomovej práce slúži na detekciu toxických komentárov v textoch s využitím neurónových sietí. Program pozostáva z jupyter notebookov (.ipynb), ktoré je možné spúšťať cez platformu **Google Colab, Datalab a Visual Studio**. Kód je rozdelený tak, aby každý notebook predstavoval kompletný experiment s jedným typom modelu (napr. GRU, BERT, ByT5).

Každý skript obsahuje:

- Načítanie a predspracovanie dát
- Definíciu architektúry konkrétneho modelu
- Trénovanie modelu
- Vyhodnotenie výsledkov pomocou metrík (accuracy, F1, recall, precision)
- Vizualizáciu výsledkov

2 Súpis obsahu dodávky

Používateľ spolu s dokumentáciou obdrží nasledujúce súbory:

- CSV súbory so surovými textovými dátami:
  - `train.json`, `test.json`
  - `slovak.xlsx`
- Súbory s predspracovaním a experimentami
- Používateľská príručka (tento dokument)
- Systémová príručka

3 Popis vývojových prostredí

V tejto kapitole sú popísané tri hlavné vývojové prostredia, ktoré boli použité v rámci tejto diplomovej práce: Google Colab, Datalab a Visual Studio Code. Každé z nich ponúka iné výhody a možnosti pre prácu s modelmi strojového učenia.

3.1 Google Colab

Google Colab je cloudové vývojové prostredie umožňujúce spúšťanie kódu v jazyku Python s podporou GPU a TPU. Umožňuje prácu bez potreby lokálnej inštalácie Pythonu a podporuje priamu integráciu s Google Diskom.

```python
import torch
import torch.nn as nn
import torch.optim as optim
import pandas as pd
from tqdm import tqdm
from torch.utils.data import Dataset, DataLoader
from transformers import MT5Tokenizer, MT5ForConditionalGeneration
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix

# Google Drive
from google.colab import drive
drive.mount('/content/drive')

# Zariadenie
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
print(f"Používam: {device}")

# tokenizer
tokenizer = MT5Tokenizer.from_pretrained("google/mt5-small")

# nacitanie dat
data_path = "/content/drive/MyDrive/combined_dataset.csv"
data = pd.read_csv(data_path)
data.columns = data.columns.str.strip()
data = data.dropna(subset=["text", "label"])
data["label"] = data["label"].astype(str)  # mT5 pracuje s textovými labelmi

# rozdelenie dat
train_texts, test_texts, train_labels, test_labels = train_test_split(
    data["text"], data["label"], test_size=0.2, random_state=42, stratify=data["label"]
)

# Dataset
class MT5TextDataset(Dataset):
    def __init__(self, texts, labels, tokenizer, max_len=128):
        self.texts = texts.tolist()
        self.labels = labels.tolist()
        self.tokenizer = tokenizer
        self.max_len = max_len
```

Obr. 3 – 1 Vývojové prostredie colab

Výhody

- Cloudové prostredie dostupné z prehliadača.
- Podpora GPU a TPU bezplatne.
- Možnosť jednoduchého zdieľania a spolupráce.
- Integrácia s Google Drive pre prístup k súborom.

Použitie v projekte

- Spúšťanie jednotlivých experimentov (napr. GRU, BERT, ByT5).
- Ukladanie a načítavanie výsledkov z Google Drive.
- Vizualizácia priebehu trénovania modelov.

Príklad pripojenia Google Disku

```
from google.colab import drive
drive.mount('/content/drive')
```

3.2 Datalab

Datalab je open-source webová aplikácia, ktorá umožňuje vytvárať a zdieľať dokumenty obsahujúce živý kód, vizualizácie a poznámky. Beží lokálne na počítači a je vhodná pre rýchle prototypovanie a vizualizáciu.

Výhody

- Spúšťanie Python kódu priamo v prehliadači.
- Interaktívne výstupy – tabuľky, grafy, obrázky.
- Kombinácia kódu a poznámok v jednom dokumente.

Použitie v projekte

- Testovanie funkcií a spracovanie dát.
- Trénovanie modelov na menších datasetoch.
- Vizualizácia výsledkov klasifikácie.

Požiadavky a spustenie

- Inštalácia cez Anacondu alebo `pip install notebook`.
- Spustenie pomocou:

```
jupyter notebook
```

3.3 Visual Studio Code

Visual Studio Code (VS Code) je multiplatformový editor s množstvom rozšírení, vhodný na vývoj komplexných Python skriptov a prácu s väčšími projektami.

```python
from sklearn.metrics import classification_report, confusion_matrix, ConfusionMatrixDisplay
import matplotlib.pyplot as plt
import torch
from torch.nn.functional import softmax

def evaluate_bert_model(model, test_loader, device, test_texts, tokenizer):
    model.eval()
    y_true = []
    y_pred = []
    texts_output = []

    with torch.no_grad():
        for i, batch in enumerate(test_loader):
            input_ids = batch["input_ids"].to(device)
            attention_mask = batch["attention_mask"].to(device)
            labels = batch["labels"].to(device)

            outputs = model(input_ids=input_ids, attention_mask=attention_mask)
            probs = softmax(outputs, dim=1)
            predictions = torch.argmax(probs, dim=1)

            y_true.extend(labels.cpu().numpy())
            y_pred.extend(predictions.cpu().numpy())

            start = i * test_loader.batch_size
            end = start + len(predictions)
            batch_texts = test_texts[start:end]

            for text, true_label, pred_label, prob in zip(
                batch_texts,
                labels.cpu().numpy(),
                predictions.cpu().numpy(),
                probs.cpu().numpy()
            ):
                if len(texts_output) < 100:
                    confidence = prob[pred_label]
                    tokens = tokenizer.tokenize(text)
                    token_ids = tokenizer.convert_tokens_to_ids(tokens)
                    texts_output.append((text, true_label, pred_label, confidence, tokens, token_ids))

    # Vypis metrik
    print("\n Klasifikačná správa:")
    print(classification_report(y_true, y_pred, digits=4))
```

Obr. 3 – 2 Vývojové prostredie Visual Studio Code

Výhody

- Výkonné rozšírenia pre Python, Git, Docker a Jupyter.
- Zabudovaný terminál a debugger.
- Možnosť pracovať so súbormi `.py` aj `.ipynb`.

Použitie v projekte

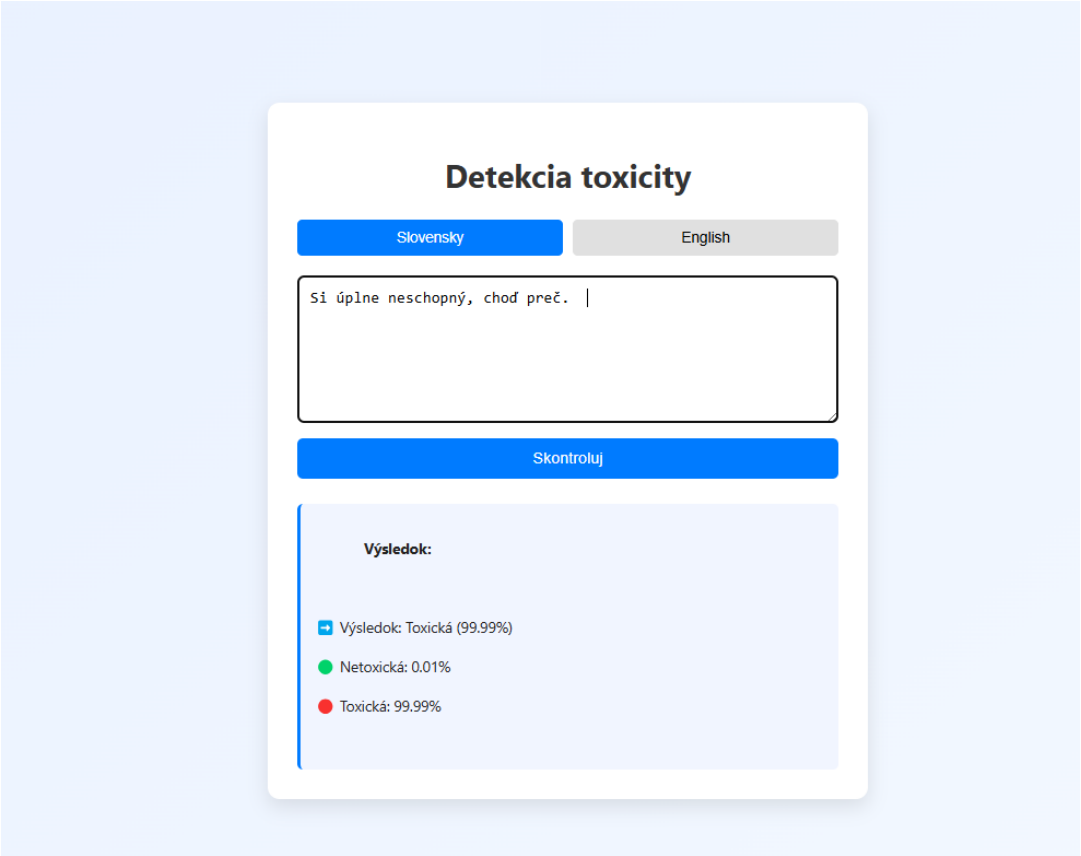
- Vývoj hlavných trénovacích skriptov.
- Úprava a testovanie modelov mimo notebookov.
- Git verzovanie celého projektu.

Požiadavky a rozšírenia

- Visual Studio Code
- Rozšírenia: **Python**, **Jupyter**, **Git**.

3.4 Nasadenie na web

Po vytvorení a otestovaní modelu nasledoval krok nasadenia modelu na webové rozhranie. Tento proces umožňuje používateľom nahrať textový vstup alebo komentár, ktorý je následne klasifikovaný modelom ako toxický alebo netoxický.



Obr. 3 – 3 Webové rozhranie

Použité technológie

- **PHP** – backendový jazyk, ktorý načítava trénovaný model a odosiela požiadavky na inferenciu.
- **Python skript** – samostatne spustiteľný skript, ktorý načíta model a vykoná predikciu.
- **HTML/CSS/JavaScript** – jednoduché rozhranie pre používateľský vstup.
- **Webhosting/Websupport** – nasadenie webovej aplikácie na server.

Postup nasadenia

1. Export trénovaného modelu do súboru (.pth alebo .bin).

2. Vytvorenie Python skriptu s načítaním modelu a predikciou podľa vstupného textu a jazyka.

3. Vytvorenie PHP skriptu, ktorý prijíma vstup od používateľa a volá Python skript pomocou `shell_exec()`.

4. Príkaz vyzerá nasledovne:

```
$command = "\"C:\\Users\\Dell\\AppData\\Local\\Programs\\Python\\Python312\\
```

5. Výstup z Python skriptu sa načíta a zobrazí používateľovi.

Výsledok

Webové rozhranie umožňuje zadať text, ktorý je po kliknutí na tlačidlo „Skontroluj“ odoslaný PHP skript. Ten následne spustí Python skript s parametrom, načíta model, vyhodnotí vstup a zobrazí výsledok (napr. „toxický“ alebo „netoxický“).

4 Systémové riešenie webového rozhrania

V rámci diplomovej práce bolo vytvorené jednoduché webové rozhranie, ktoré slúži ako nadstavba nad trénovaný model neurónovej siete. Umožňuje používateľovi nahrať vlastný text a získať predikciu toxicity v reálnom čase.

Architektúra systému

Systém je rozdelený na dve hlavné časti:

- **Backend (PHP + Python)** – PHP skript prijíma požiadavku, spúšťa Python skript s modelom a vracia výsledok.
- **Frontend (klient)** – HTML stránka s jednoduchým formulárom na odosielanie textu.

Princíp fungovania

1. Používateľ zadá text do poľa na webstránke.
2. Text sa odošle metódou POST PHP skriptu.
3. PHP skript zostaví príkaz na spustenie Python skriptu s argumentmi.
4. Python skript načíta model a vykoná klasifikáciu.
5. Výsledok sa vráti a zobrazí na stránke.

Výhody riešenia

- Jednoduché nasadenie na klasický webhosting (bez potreby Python servera).
- Prepojenie PHP rozhrania s výkonným modelom v Pythone.
- Možnosť prispôsobenia výsledku a vzhľadu webu.