VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

Soft Computing

Technická zpráva

2025 Samuel Kiszka (xkiszk00)

Úvod

Vytvořený model se zabývá úlohou binární klasifikace sentimentu filmových recenzí z datasetu IMDB.

Cílem je rozhodnout, zda je textová recenze pozitivní, nebo negativní.

Model využívá kombinaci transformeru klasifikátoru implementovaného v knihovně NumPy a předtrénovaného modelu z knihovny Hugging Face, který text před vstupem do vlastního modelu převádí na kontextový embedding.

Během trénování byla porovnávána výkonnost modelu při použití různě velkých datasetů a také rozdíl mezi čistě NumPy implementací a referenční verzí v PyTorch.

Teorie

Transformer

Transformer je architektura neuronových sítí určená pro zpracování sekvencí, v mém případě přirozeného jazkya, která používá mechanismem *self-attention*. Ten umožňuje modelu zohlednit vzájemné vztahy mezi všemi tokeny vstupní sekvence současně.

V tomto projektu je využita pouze **encoder část Transformeru** pro zpracování embeddingu vstupního textu. Předtrénovaný model (DistilBERT) převede textovou recenzi na vektorovou reprezentaci, tu poté dostane na vstup Encoder, který vstup zanalyzuje a pochopí jeho obsah a význam v celém kontextu, a slouží jako vstup pro klasifikátor,.

Optimalizátor AdamW

Jako optimalizační algoritmus jsem použil AdamW, který je vylepšením klasického optimalizátoru Adam. Přidává regularizaci *weight decay*, která zlepšuje generalizaci modelu.

Princip fungování:

- Sleduje pohyb průměrné hodnoty gradientu a jeho druhé momenty (*moving averages*).
- Adaptivně upravuje velikost kroků pro každý parametr zvlášť podle historie gradientů.
- Weight decay odděluje regularizaci od samotného kroku optimalizace, což vede k lepší stabilitě učení.

Aktualizace parametrů pomocí AdamW lze vyjádřit vzorcem:

$$
\begin{aligned}
m_t &= \beta_1 m_{t-1} + (1-\beta_1) g_t, \\
v_t &= \beta_2 v_{t-1} + (1-\beta_2) g_t^2, \\
\hat{m}_t &= \frac{m_t}{1-\beta_1^t}, \quad \hat{v}_t = \frac{v_t}{1-\beta_2^t}, \\
\theta_{t+1} &= \theta_t - \eta \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon} - \eta \lambda \theta_t,
\end{aligned}
$$

kde:

- θ_t jsou parametry modelu v kroku t,
- g_t je gradient ztrátové funkce vzhledem k parametrům,
- m_t, v_t jsou exponenciální klouzavé průměry prvního a druhého momentu gradientu,
- β_1, β_2 jsou hyperparametry určující váhu historie gradientů (v mé implementaci 0.9 a 0.999),
- $\hat{m}_t, \hat{v}_t$ jsou korekce biasu na začátku tréninku,
- η je učící krok (learning rate) (defaultně 2e-5),
- λ je koeficient weight decay (v mé implementaci 1e-4),
- ϵ malá konstanta pro stabilitu dělení.

Architektura

Architektura modelu se skládá z několika částí:

1. **Embeddingová vrstva** – využívá předtrénovaný jazykový model (DistilBERT), který z textu vytvoří vektorovou reprezentaci celé věty. Jako výstup se bere celá recenze.
2. **Transformer Encoder** – Nad výstupem embedingové vrstvy provádí transformaci vstupního tenzoru pomocí definovaného počtu vrstev transformeru, kde v každé je zapojen definovaný počet hlav na analýzu vstupních dat. **Implementováno v NumPy.**
3. **Klasifikátor** – jednoduchá plně propojená vrstva:

$$y = \text{softmax}(Wx + b),$$

   která vrací pravděpodobnosti pozitivního a negativního sentimentu. **Implementováno v NumPy.**

Pro učení modelu byl použit optimalizační algoritmus **AdamW** popsaný výše.
Kód je rozdělen do následujících souborů:

- `train.py` – hlavní trénovací skript,
- `src/data_loader.py` – načítání a předzpracování datasetu IMDB pomocí předtrénovaného modelu,
- `src/transformer_encoder.py` – definice NumPy modelu - výsledný model `TransformerClassifier` se skládá ze dvou podmodulů - `Encoder`, transformer encoder model provádějící multihead attention a `Classifier` provádějící klasifikaci sentimentu na základě výstupu encoderu,
- `src/trainer.py` – implementace trénování jednotlivých epoch, definice optimalizátorů a vyhodnocování úspěšnosti trénování,
- `src/train_torch.ipynb` – referenční model implementovaný v PyTorch
- `review_classification.py` – výsledný skript, který pomocí natrénovaného modulu provádí inferenci sentimentu nad interaktivně zadávanými daty.

Během trénování se automaticky vytvářejí následující složky:

- `data/` – cache datasetu, aby nebylo v případě opakovaného trénování nutné data znovu stahovat,
- `models/` – uložené parametry trénovaných modelů - vždy nejúspěšnější varianta s danými vstupními parametry,
- `results/` – grafy trénovacích a validačních metrik na základě trénování.

Výsledky trénování

Implementace umožňuje nastavit modul na trénovaní s různým počtem vrstev, attention hlav a velikostí feed forward vrstvy, ale samotné trénování proběhlo u všech testovaných modelů s následujícími parametry:

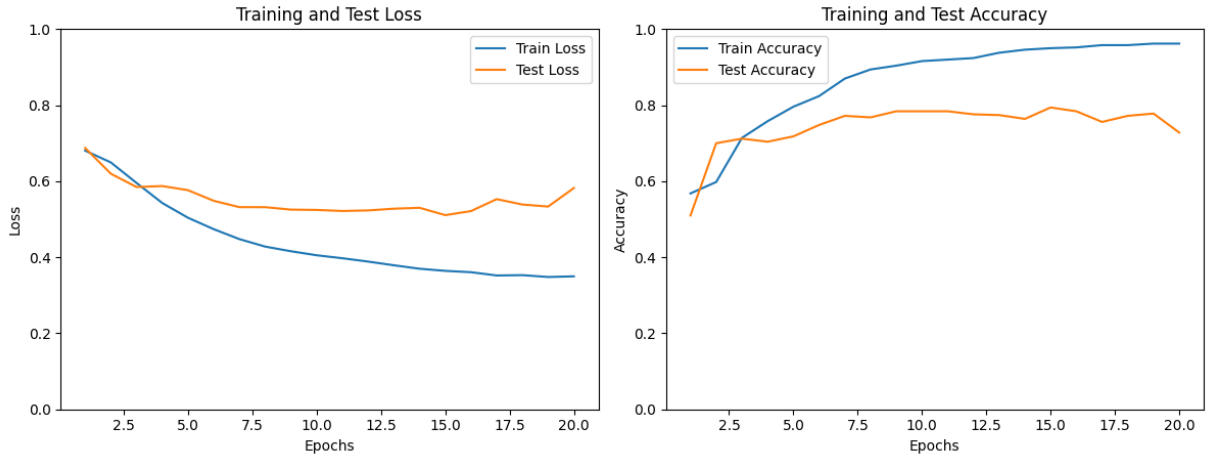
Atribut	Počet vrstev	Počet hlav na vrstvu	FF neurony	Batch size
Hodnota	2	4	64	32

Modely byly trénovány nad různě velikými vstupy, přičemž ve vstupních datech jsou vždy rovnoměrně reprezentovány obě třídy, pro porovnání výsledků. Ze stejného důvodu bylo trénování provedeno také na referenčním modelu implementovaném v PyTorch.

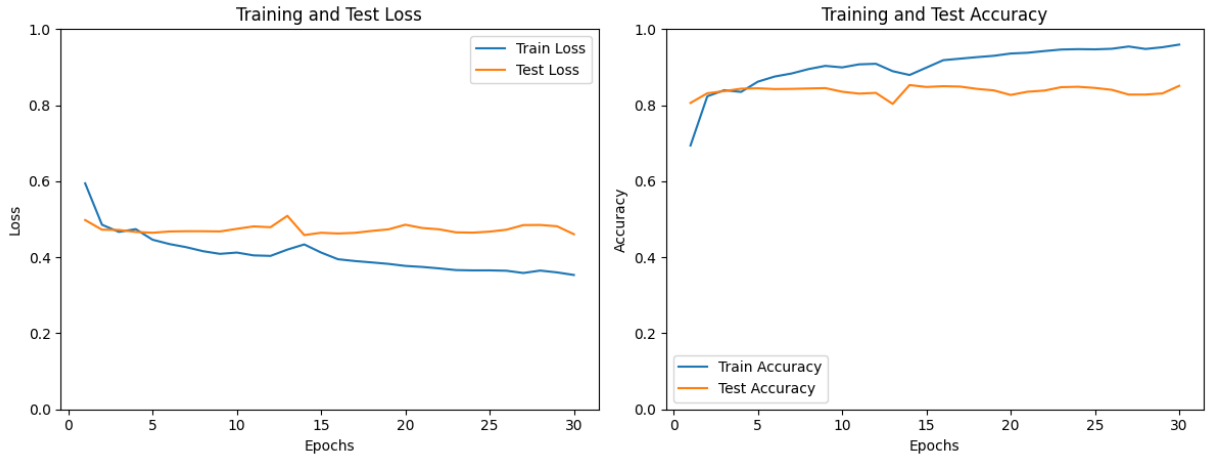
Model	Počet vstupů	Počet epoch	Čas trénování	Přesnost na testu
Model1	500	20	80 min	79%
Model2	2000	30	470 min	85%
PyTorch	1000	20	6 min	87 %

Tabulka 1: Porovnání trénování modelu nad různě velikými daty

Na následujících grafech lze vidět postupný vývoj trénovacích metrik pro Model1 a Model2 z tabulky výše.



Obrázek 1: Průběh trénovacích metrik během učení modelu s 500 vstupy.



Obrázek 2: Průběh trénovacích metrik během učení modelu s 2000 vstupy.

Poměrně dobré výsledky vzhledem k velmi malému trénovacímu datasetu lze pravděpodobně přisuzovat především použité embeddingové vrstvě, která již sama vstup předzpracovává a na vstup samotného transformeru se dostává již poměrně kvalitní kontextová reprezentace. Pro dosažení podobných výsledků bez použití této embedingové vrstvy by bylo nutné mnohem větší množství dat, na které nemám výpočetní kapacitu.

Manuály

Projekt je implementován v jazyce Python a pro ten je nejjednodušší vytvořit virtuální prostředí, ve kterém se nainstalují všechny potřebné balíčky, a které lze poté jednoduše smazat. S tímto také přistupuji k následujícím manuálům pro připravení prostředí a následné spuštění projektu. Manuály také předpokládají, že soubor s implementací projektu je již přítomný ve virtuálním stroji a že následující příkazy jsou spouštěny z kořene projektového adresáře.

Příprava prostředí

Jelikož poskytnutý virtuální stroj SFC defaultně neobsahuje balíčky potřebné pro vytvoření pythonovského virtuálního prostředí, je tyto nutné doinstalovat. A jelikož se jedná o instalaci systémových balíčků, je potřeba ji spouštět s administrátorským oprávněním. Instalační skript pro stažení potřebných balíčků se nachází ve složce `scripts` a níže je uvedený příkaz pro jeho exekuci. Tento skript také rovnou vytvoří pythonovské virtuální prostředí ve složce `.venv`.

```
sudo bash ./scripts/setup_sfc_env.sh
```

Po instalaci systémových balíčků je nutné manuálně spustit vytvořené virtuánlí prostředí a následně stáhnout všechny potřebné pythonovské balíčky pomocí `requirements.txt`.

```
source .venv/bin/activate
pip install -r requirements.txt
```

Příprava prostředí může zabrat nějaký čas, v závislosti na kvalitě internetového připojení. Stahují se nicméně velké pythonovské balíčky, které mají spoustu vlastních závislostí, které se rekurzivně stáhnou také.

Spuštění tréninku

Když je prostředí připravené, je možné spustit samostatný trénink modelu. To samo o sobě trvá poměrně dlouho, v závislosti na zvolených parametrech a proto je možné tento krok přeskočit a stáhnout model, který jsem natrénoval doma a uložil na internetové uložiště nextcloud. K tomu slouží následující skript:

```
. ./scripts/download_default_model.sh
```

Jedná se o **Model2** z tabulky výše.

Případně je možné spustit trénink modelu s vlastní volbou parametrů. Lze si zvolit `počet vrstev`, `počet hlav na vrstvu`, `počet trénovacích epoch`, `velikost trénovacího datasetu`, `dimenzi FF vrstvy`, `velikost batche` a `učicí konstatnu`. Se zvolenými parametry lze poté spustit učení pomocí pythonovského skriptu:

```
python train.py
```

Tyto parametry lze zadat při spuštění skriptu pomocí klasických přepínačů a formát parametrů a jejich defaultní hodnoty, které budou použity, nebudou li specifikované jiné, lze nahlédnout pomocí přepínače `--help`:

```
python train.py --help
```

První spuštění tréninku může trvat o něco déle, jelikož se musí stáhnout IMDB dataset a embedingový model. Tyto se ale cachují a následné spouštění již opakované stažení nevyžaduje. Na základě zadaného počtu trénovacích vstupů (který se zadává jako počet vstupů jedné třídy a reálný počet trénovacích vstupů je tedy dvojnásobný) se ale musí pomocí embeddingového modelu vytvořit subset IMDB datasetu používaný pro trénink modelů s danou velikostí vstupu. Tyto se v případě opakovaného spouštění nad stejně velkým datasetem znovu přepoužívají.

Použití modelu pro inferenci

Pro použití natrénovaného modelu ke klasifikaci poté slouží druhý skript `review_classificatino.py`. I tento bere na vstupu několik parametrů: `cestu k modelu`, `počet vrstev modelu`, `počet hlav na vrstvu` a `dimenzi FF vrstvy`. Je nutné, aby model byl přítomný na určené cestě a aby zmíněné parametry odpovídaly parametrům modelu, který je na dané cestě uložen. V opačném případě spadne skript s errorem. Defaultně je skript nastaven pro práci se staženým modelem a bez parametrů bude tedy pracovat s ním. Je tedy nutné před jeho spuštěním model stáhnout pomocí skriptu zmíněného výše. Poté je možné spustit program:

```
python review_classification.py
```

Program je interaktivní a dovoluje postupně zapisovat nové recenze a jejich ohodnocení a predikuje jejich ohodnocení na základě modelu. Po ukončení vypíše přesnost inference aktuálního běhu.

Ukázka běhu:

```
Model parameters loaded from models/default_model.npz

Enter a review and class divided by '|' (type 'exit' to quit):
        Positive example: This movie was fantastic! I loved it. | 1
        Negative example: I hated this movie. It was terrible. | 0
> What a wonderful movie! | 1
Predicted Sentiment: Positive
> I relly liked this movie. It was inspirational and I took lot from it. | 1
Predicted Sentiment: Positive
> This film was long and boring. Wasted money | 0
Predicted Sentiment: Negative
> I'd like to meet someone, who could have liked this film.. | 0
Predicted Sentiment: Positive
> I would not recomend this movie. It was so predictable. | 0
Predicted Sentiment: Negative
> exit
Overall Accuracy: 80.00%
```