VŠB TECHNICKÁ UNIVERZITA OSTRAVA
FAKULTA ELEKTROTECHNIKY A INFORMATIKY

Detekce strojově přeložených textů pomocí strojového učení

Machine learning detection of machine translated texts

Bc. Jan Kusák

Diplomová práce

Vedoucí práce: prof. Ing. Jan Platoš, Ph.D.

Ostrava, 2023

VŠB TECHNICKÁ UNIVERZITA OSTRAVA | FAKULTA ELEKTROTECHNIKY A INFORMATIKY

Č.j.VSB/23/047670
vsboee2301433E

Zadání diplomové práce

Student: **Bc. Jan Kusák**

Studijní program: N2647 Informační a komunikační technologie

Studijní obor: 2612T025 Informatika a výpočetní technika

Téma: Detekce strojově přeložených textů pomocí strojového učení
Machine Learning Detection of Machine Translated Texts

Jazyk vypracování: čeština

Zásady pro vypracování:

Cílem práce je popsat a navrhnout klasifikátor pro textová data, který bude schopen detekovat strojově přeložené texty pomocí Google Translate a DeepL. Tento klasifikátor realizovat pomocí vhodných technologií a ověřit jeho funkčnosti vzhledem ke zvolenému datasetu.

Práce bude obsahovat:
1. Popis problematiky klasifikace textových dat.
2. Návrh vhodného klasifikátoru nebo více klasifikátorů.
3. Implementace klasifikátory pomocí vhodných technologií a frameworků.
4. Ověření fungování navrženého postupu a porovnání výsledků s jinými metodami.

Seznam doporučené odborné literatury:

[1] VASWANI, Ashish, Noam SHAZEER, Niki PARMAR, Jakob USZKOREIT, Llion JONES, Aidan N. GOMEZ, Łukasz KAISER a Illia POLOSUKHIN. Attention is all you need. In: Advances in neural information processing systems. 2017, s. 5998-6008.
[2] ROTHMAN, Denis. Transformers for natural language processing: build innovative deep neural network architectures for NLP with Python, PyTorch, TensorFlow, BERT, RoBERTa, and more. Birmingham: Packt Publishing, 2021. Expert insight. ISBN 978-1800565791.
[3] ATIENZA, Rowel. Advanced Deep Learning with TensorFlow 2 and Keras: Apply DL, GANs, VAEs, deep RL, unsupervised learning, object detection and segmentation, and more. 2nd edition. Packt Publishing, 2020. ISBN 978-1800568273.

17. listopadu 2172/15
708 00 Ostrava-Poruba
Česká republika

spojovatelka: +420 597 321 111
epodatelna: epodatelna@vsb.cz
ID datové schránky: d3kj88v

IČ: 61989100
DIČ: CZ61989100

email: studijni.fei@vsb.cz
www.fei.vsb.cz

Formální náležitosti a rozsah diplomové práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí diplomové práce: **prof. Ing. Jan Platoš, Ph.D.**

Datum zadání: 01.09.2022

Datum odevzdání: 04.07.2023

Garant studijního oboru: prof. RNDr. Václav Snášel, CSc.

V IS EDISON zadáno: 02.06.2023 08:32:23

Abstrakt

Cílem práce je podrobně popsat a navrhnout klasifikátor pro textová data, který bude schopný identifikovat strojově přeložené texty pomocí služeb Google Translate a DeepL. Pro dosažení tohoto cíle je nutné vytvořit vlastní dataset, na kterém budou různé architektury a modifikace modelů trénovány a následně testovány.

Klíčová slova

Strojové učení, Neuronové sítě, Transformer, LSTM, GRU, BERT, NLP, Sémantická analýza, Dataset, Model

Abstract

This thesis aims to comprehensively describe and design a text data classifier capable of detecting machine-translated texts that have been translated via Google Translate and DeepL. This task necessitates the creation of a customized dataset, on which various architectures and modifications of models will be trained and subsequently evaluated.

Keywords

Machine learning, Neural Network, Transformer, LSTM, GRU, BERT, NLP, Semantic analysis, Dataset, Model

Poděkování

Rád bych poděkoval panu prof. Ing. Janu Platošovi, Ph.D. který mě uvedl do světa strojového učení. Ještě při mé bakalářské práci mě vzal pod svá křídla a otevřel mi dveře do této populární a již mnou oblíbené oblasti. Velmi si vážím jeho trpělivosti, střídmosti a množství cenných rad, které mi poskytl. Rovněž bych chtěl vyjádřit velké díky své rodině za veškerou podporu a motivaci. Také bych rád poděkoval svým přátelům za pomoc s korekturou a opravami mé diplomové práce.

Obsah

Seznam použitých zkratek a symbolů

NN	–	Neuronová síť
DNN	–	Hluboká neuronová síť
RNN	–	Rekurentní neuronová síť
LSTM	–	Long Short-Term Memory
GRU	–	Gated Recurrent Unit
Transformer	–	Transformer neuronová síť
BERT	–	Bidirectional Encoder Representations from Transformers
NLP	–	Zpracování přitozeného jazyka
GloVe	–	Global Vectors for Word Representation

Seznam obrázků

Seznam tabulek

Kapitola 1

Seznámení

V dnešní společnosti je překlad mezi různými jazyky klíčovým nástrojem pro komunikaci a sdílení informací. S rychlým pokrokem technologií v oblasti strojového překladu se stává stále běžnějším používat automatické systémy pro překlad textu. Tyto systémy, založené na algoritmech strojového učení, jsou schopny generovat překlady s překvapivou úrovní přesnosti. Nicméně, přestože se technologie strojového překladu neustále zdokonalují, stále existuje problém s detekcí strojově přeložených textů. V některých případech může být důležité rozlišovat mezi lidským překladem a strojovým překladem. Cílem tohoto projektu je vytvořit efektivní a spolehlivý model pro detekci strojově přeložených textů pomocí technik strojového učení. Práce se zabývá detekcí nad lidsky a strojově psanými textovými soubory, které se použijí jako vstupní data pro detekční model. Následně se budou vyvíjet a trénovat algoritmy strojového učení, které budou schopny rozpoznat specifické znaky strojově přeloženého textu a odlišit je od lidského překladu. Výsledkem tohoto projektu bude vytvoření detekčního modelu, který bude schopen s přijatelnou přesností identifikovat strojově přeložené texty.

První kapitola se věnuje problematice zpracování přirozeného jazyka (NLP). Zabývá se konkrétně sentimentální analýzou a popisuje následné techniky pro převod textu do strojově srozumitelného formátu. V první části je podrobněji vysvětlena sentimentální analýza. Následně se kapitola zaměřuje na techniky převodu textu do strojově srozumitelného formátu. Těmito technikami se rozumí proces transformace přirozeného jazyka na formu, kterou může snadno zpracovat počítač či jiný stroj. Tato kapitola poskytuje ucelený přehled sentimentální analýzy a technik převodu textu do strojově srozumitelného formátu, které jsou využívány v oblasti zpracování přirozeného jazyka.

Druhá kapitola stručně popisuje koncept neuronových sítí. Dále se věnuje modifikacím neuronových sítí, které jsou známy jako rekurentní neuronové sítě (RNN), a jejich vylepšením. Tato druhá kapitola poskytuje základní přehled o neuronových sítích a jejich využití v kontextu rekurentních neuronových sítí a jejich vylepšení.

V třetí kapitole je popsána jedna z nejmodernějších forem neuronových sítí - síť Transformers. Dále zdůrazňuje její význam v oblasti zpracování přirozeného jazyka a dalších úloh spojených se

sekvencemi dat.

Předposlední kapitola poskytuje detailní přehled o výsledcích praktické části, kde byly zkoumány různé architektury modelů a jejich konfigurace. Tyto výsledky jsou klíčové vyhodnocení úspěšnosti a vhodnosti jednotlivých modelů pro daný problém.

V poslední kapitole je poskytnuto shrnutí dosažených výsledků natrénovaných modelů a navrhnuty možnosti jejich zlepšení.

Kapitola 2

Zpracování přirozeného jazyka

V této práci bylo nezbytné zajistit, aby stroj byl schopen alespoň částečně porozumět lidsky psanému textu. Přesně touto problematikou se zabývá oblast Zpracování přirozeného jazyka (Natural language processing - NLP) [1] [2], která kombinuje různé techniky, jako je lingvistika, informatika a umělá inteligence, s cílem zkoumat interakci mezi strojem a lidským jazykem. S využitím poznatků z této oblasti je možné programovat počítače k zpracování a analýze velkých množství textových dat. Tím se umožňuje počítačům porozumět kontextu v textu. Zpracování přirozeného jazyka se zaměřuje na širokou škálu výzev, včetně rozpoznávání řeči, porozumění přirozenému jazyku a generování přirozeného jazyka. Má bohatou historii a vyvinulo se mnoho technik pro zpracování přirozeného jazyka. V této konkrétní práci je zvláště vhodná technika nazývaná Sentimentální analýza (Sentiment Analysis), která vychází z přirozeného zpracování jazyka.

2.1 Sentimentální analýza

Sentimentální analýza (Sentiment analysis) [3] je jednou z nejčastěji používaných technik v oblasti zpracování přirozeného jazyka. Je široce využívána pro řešení problémů spojených s určováním polarity textu, tj. zda je text pozitivní, neutrální nebo negativní. Tato technika se aplikuje na různé typy textů, jako jsou dotazníky, komentáře, recenze a také na detekci strojového překladu, což je také cílem této práce. Před použitím analýzy je však nezbytné předzpracovat text, na kterém chceme provádět detekci, a převést ho do formátu, který je srozumitelný pro modely.

2.2 Metody analýzy textu

Algoritmy strojového učení, jako jsou ty využívané při zpracování přirozeného jazyka, preferují jasně definovaná vstupní data pevné délky. Proto je často potřeba přizpůsobit data tak, aby odpovídala požadovanému formátu. To může zahrnovat osekávání nebo doplňování prázdnými hodnotami. Tyto algoritmy nejsou schopny přímo pracovat s textem ve své původní podobě. Z tohoto důvodu je

nezbytné provést předzpracování lidského textu a převést jej na číselnou podobu, například pomocí vektorizace textu. Existuje mnoho různých metod a technik, které slouží k tomuto účelu.

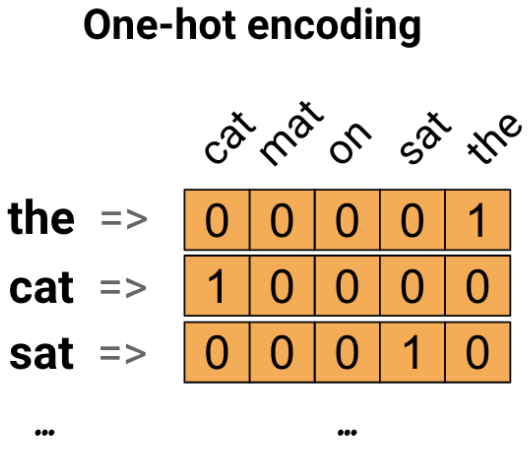
2.3 One-hot kódování

Metoda One-hot kódování (One-hot encoding) je základní technikou převodu textu do číselné podoby v oblasti zpracování přirozeného jazyka. Tato metoda pracuje s tabulkou, nazývanou slovníkem, která mapuje jedinečná slova v textu na binární vektory. Každé unikátní slovo ve větě je reprezentováno nulovým vektorem předem definované velikosti, a na pozici odpovídající konkrétnímu unikátnímu slovu je nastavena hodnota 1. Tato hodnota indikuje přítomnost daného slova v číselném vektoru. Tímto způsobem je možné převést lidsky psané texty do číselné podoby, kterou následně mohou různé algoritmy zpracovávat.

2.3.1 Nevýhody One-hot kódování

Metoda One-hot 2.1 kódování má významné omezení, kterým je značná neefektivnost pro velké množství unikátních slov. Představme si větu obsahující tisíc slov. Pro každé unikátní slovo v této větě by bylo potřeba vytvořit unikátní nulový vektor, kde pouze jedno místo by mělo hodnotu 1 pro reprezentaci konkrétního slova. To znamená, že většina hodnot ve vektoru by byla nulová, což je značné plýtvání pamětí a výpočetních zdrojů. Navíc, s přidáním každého nového slova se musí zvětšovat velikost všech vektorů ve slovníku a přidávat hodnoty 0, které nemají žádný význam.

Tento problém je jedním z důvodů, proč byly vyvinuty efektivnější metody reprezentace textu, jako je například vkládání slov (word embedding).



Obrázek 2.1: One-hot encoding [4]

2.3.2 Využití v praxi One-hot kódování

V následující ukázce 2.1 je prezentována implementace metody pro konverzi lidsky psaného textu do číselných vektorů pomocí one-hot kódování.

```
# Převod textu pomocí One-hot kódování
lidsky_text = [
   "Ja jsem nejlepsi ze vsech",
    "Ja jsem nejhorsi ze vsech"
   ]

unikatni_slova = set()
for veta in lidsky_text:
    for slovo in veta.split():
        unikatni_slova.add(slovo)
    ...

slovnik = {}
for unikatni_index, unikatni_slovo in enumerate(unikatni_slova):
    one_hot_encoding_vektor = [0] * len(unikatni_slova)
    one_hot_encoding_vektor[unikatni_index] = 1
    slovnik[unikatni_slovo] = one_hot_encoding_vektor
    ...
```

Listing 2.1: One-hot kódování v praxi

V uvedené ukázce je představeno vytvoření slovníku, kde každé unikátní slovo je převedeno na číselný vektor pomocí metody one-hot kódování. Níže 2.2 je příklad reprezentace několika slov ve slovníku:

```
nejhorsi : [1, 0, 0, 0, 0, 0]
ze       : [0, 1, 0, 0, 0, 0]
Ja       : [0, 0, 1, 0, 0, 0]
nejlepsi : [0, 0, 0, 1, 0, 0]
jsem     : [0, 0, 0, 0, 1, 0]
vsech    : [0, 0, 0, 0, 0, 1]
```

Listing 2.2: One-hot kódování prezentace výsledného formátu dat

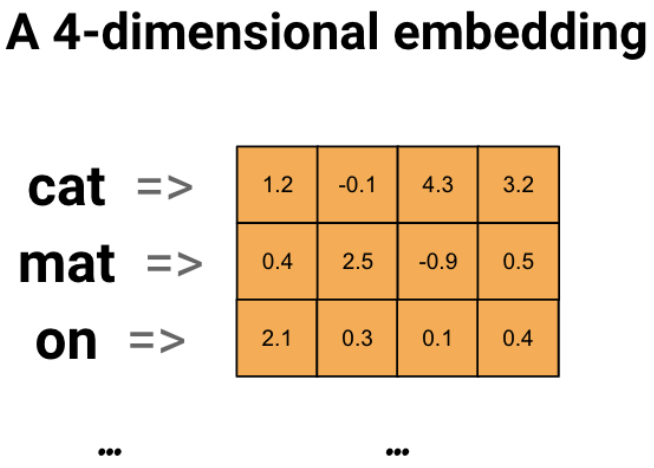
2.4 Vkládání slov

Vkládání slov (word embedding) je pokročilá metoda pro převod lidsky psaného textu do číselného vektoru, která nahrazuje one-hot kódování. Tato metoda, která se využívá i v praktické části této práce, umožňuje reprezentovat slova pomocí vektorů s desetinnými čísly. Každé číslo ve vektoru reprezentuje různé vlastnosti a vztahy mezi slovy.

Pro výpočet těchto vektorů se používají různé metody, které umožňují zachytit kontext slov a jejich vztahy. Na rozdíl od one-hot kódování, kde každé slovo má pouze jedinou hodnotu 1 ve vektoru, vkládání slov přináší bohatší informace o kontextu daného slova v textech.

Vkládání slov [4] 2.2 má také vlastnost vysoce dimenzionální reprezentace slov, pomocí které zachycuje kontext, ve kterém se slova vyskytují. Tato vylepšená metoda umožňuje strojům, zpracovávající přirozený jazyk, pracovat s dodatečnými informacemi o podobnosti slov.

Díky technice vkládání slov mají stroje větší schopnost porovnávat a porozumět podobnosti mezi slovy na základě jejich kontextu. To přináší vylepšení v oblasti zpracování přirozeného jazyka a umožňuje lépe porozumět lidsky psanému textu.



Obrázek 2.2: Word embedding [4]

2.4.1 Word2Vec

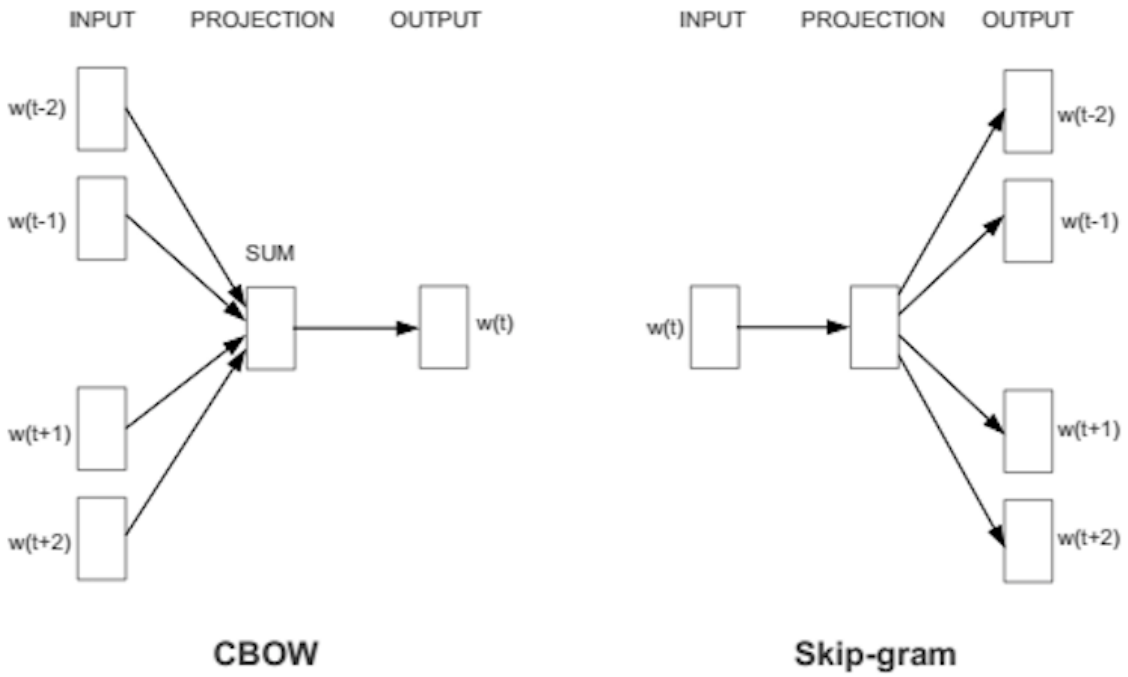
Word2Vec [5] je významným příkladem modelu pro vytváření vkládání slov [4] a byl vyvinutý Tomášem Mikolovem z Google. Tento model umožňuje efektivní převod textu do strojové podoby a zachycuje vztahy mezi slovy.

Word2Vec 2.3 se skládá z několika algoritmů, které jsou schopné naučit se reprezentaci slov. Existují dva hlavní modely pro učení techniky vkládání slov pomocí Word2Vec: Continuous Bag-of-Words (CBOW) a Continuous Skip-gram.

Continuous Bag-of-Words (CBOW) model se snaží předpovědět aktuální slovo na základě okolních slov v kontextu. Naopak, Continuous Skip-gram model se snaží na základě aktuálního slova předpovědět okolní slova v kontextu. Oba tyto modely přispěly k rozvoji výzkumu vkládání slov a převodu textu do strojové podoby.

Nicméně, i přes úspěch Word2Vec se ukázalo, že existuje efektivnější metoda, která dosahuje stejných výsledků. Tato metoda se nazývá GloVe (Global Vectors for Word Representation) [6]. GloVe je algoritmus, který kombinuje globální statistiky slov a lokální kontextové informace při vytváření vkládání slov. Tento přístup umožňuje efektivnější a výkonnější reprezentaci slov.

GloVe se stal populární volbou pro vytváření vkládání slov díky své schopnosti zachytit vztahy mezi slovy ve velkých korpusových datech. Tato metoda přinesla další pokrok v oblasti zpracování přirozeného jazyka a významně přispěla k porozumění slov a jejich kontextu strojovými modely.



Obrázek 2.3: Word2Vec představení možností [5]

2.5 GloVe

GloVe (Global Vectors for Word Representation) [6] je vylepšeným modelem v porovnání s Word2Vec [5] a také vytváří reprezentaci vkládaných slov. GloVe představuje vylepšenou verzi Word2Vec a přináší

několik změn.

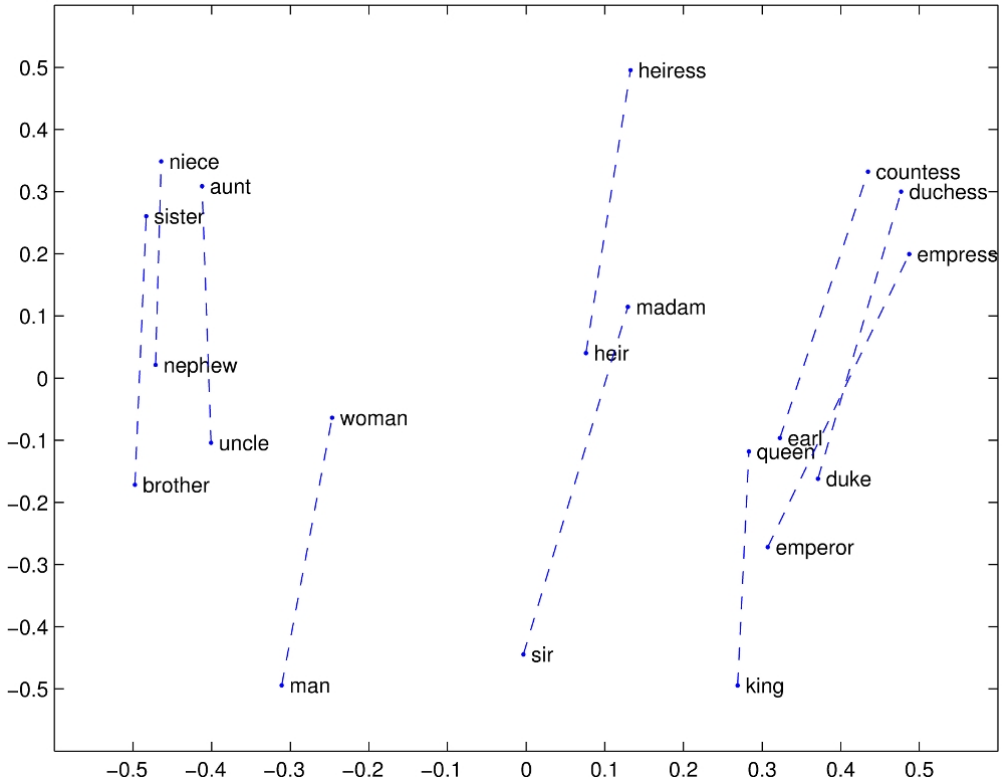
Hlavním rozdílem mezi GloVe a Word2Vec je, že GloVe kombinuje obě techniky, Continuous Bag-of-Words (CBOW) a Skip-gram, místo omezení na jednu z nich. Tímto způsobem GloVe získává výhody obou přístupů.

GloVe je založen na počtu výskytů (count-based). Tento model se učí vytvářet vkládání slov pomocí redukce dimenzí na matici počtů společných výskytů slov. Nejprve se vytvoří velká matice, ve které jsou zachyceny informace o tom, jak často se slova vyskytují v různých kontextech ve vstupních datech. Poté je tato matice převedena na matici nižší dimenze, kde každý řádek představuje vektorovou reprezentaci jednoho slova.

V případě GloVe je matice počtů normalizována vzhledem k počtu výskytů a následně upravena pomocí log-smoothing techniky. GloVe produkuje reprezentaci vkládaných slov, která je kontextově nezávislá, což znamená, že každé slovo je reprezentováno pouze jedním 1D vektorem, který sdružuje různé vlastnosti a vztahy tohoto slova ve vstupním textu. GloVe je také model založený na slovech (word-based), což znamená, že přijímá slova jako vstupní data a generuje vkládání slov jako výstup.

GloVe a Word2Vec se obecně podobně chovají a dosahují podobných výsledků. Výhodou GloVe je však snadnější implementace paralelizace, což vede k vyšší efektivitě.

Vytvořená reprezentace vztahů mezi jednotlivými slovy lze pochopitelněji vypozorovat z obrázku 2.4



Obrázek 2.4: GloVe reprezentace slov ve 2D prostoru [6]

2.5.1 Continuous Bag-Of-Words

Continuous Bag-Of-Words (CBOW) [7] je model strojového učení používaný pro vytváření reprezentace vkládaných slov. Jeho hlavním cílem je předpovědět cílové slovo na základě kontextu, který je definován jako určitý počet slov před a za daným slovem v textu. CBOW se snaží naučit vektory slov, které mají kvalitní reprezentace pro předpovídání cílových slov na základě kontextuálního významu.

2.5.2 Skip-gram

Metoda Skip-gram [8] je další volbou pro model Word2Vec a také je používána společně s CBOW v GloVe. Hlavním rozdílem mezi Skip-gram a CBOW je v tom, co tyto metody přesně předpovídají.

CBOW předpovídá slovo na základě jeho okolí. Zkouší určit, jaké slovo by se nejlépe hodilo do středu daného okolí na základě kontextu. Například, pokud bylo okolí slova 'auto' ve větě 'Rychle jedoucí auto předjelo nákladní vozidlo' CBOW se snaží předpovědět slovo 'rychle' na základě slov 'jedoucí' a 'předjelo'.

Na druhou stranu, metoda Skip-gram předpovídá okolí daného slova. Zkouší určit, jaká slova by se mohla objevit v okolí daného slova na základě kontextu. Pokračujeme-li v příkladu věty 'Rychle jedoucí auto předjelo nákladní vozidlo', Skip-gram by se snažil předpovědět slova 'jedoucí' a 'předjelo' na základě slova 'auto'.

2.5.3 CBOW a Skip-gram porovnání

V rámci modelu Word2Vec je metoda CBOW rychlejší, protože zpracovává celý kontext jako jednu entitu.

Metoda Skip-gram vytváří různé páry slov pro každé slovo v kontextu. Každý pár se skládá z jednoho vstupního centrálního slova a jednoho výstupního okolního slova. Skip-gram se snaží předpovědět okolní slova na základě daného centrálního slova.

Nicméně, výběr mezi metodou CBOW a Skip-gram závisí na konkrétních úkolech a datových sadách. Obě metody mají své výhody a omezení a jejich volba závisí na konkrétních potřebách a podmínkách aplikace.

Kapitola 3

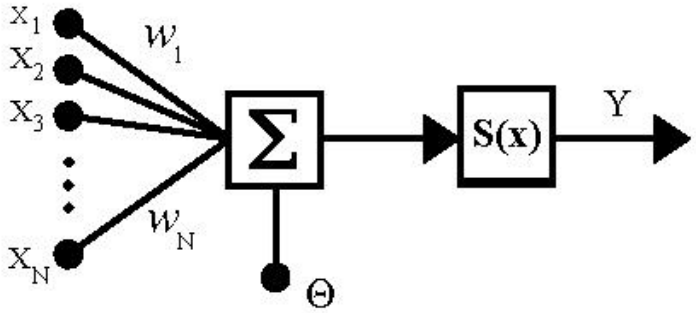
Neuronové sítě

Neuronové sítě (Neural Network) [9] [10] jsou výpočetní modely inspirované biologickým mozkem. Základními stavebními jednotkami neuronové sítě jsou neurony, které mají své vstupy, váhy a výstup.

Každý neuron obdrží vstupní data od jiných neuronů, přičemž každému vstupu je předána určitá váha. Váhy slouží k určení důležitosti jednotlivých vstupů. Neuron zpracovává své vstupní data pomocí aktivačních funkcí a provede sumaci vážených vstupů. Výsledek této operace je pak předán dalším neuronům jako výstup.

Je důležité poznamenat, že neuronový model 3.1 může mít libovolný počet vstupů a váh, ale každý neuron generuje pouze jeden výstup. Tímto způsobem se vytváří komplexní sítě neuronů, které spolupracují při zpracování informací a řešení problémů.

Neuronové sítě se využívají v mnoha oblastech umělé inteligence a strojového učení. Jejich schopnost zpracovávat složité vzorce a adaptovat se na data je důležitá pro úspěšné řešení rozmanitých úloh, jako je rozpoznávání obrazu, překlad textu, predikce a další.



Obrázek 3.1: Model neuronu [9]

V následujícím vzorci je formule podle McCulloch–Pitts modelu neuronu. Váhy se zde násobí společně se vstupními daty, které se následně sčítají a upravují podle parametrizované hodnoty

tzv. biasu. Výsledek se vkládá do aktivační funkce a vychází výstup, který se posílá na vstup následujícímu neuronu.

$$Y = S(\sum_{i=1}^{N}(W_i X_i) + \Theta)$$

- X: vstupní data
- W: váhy mezi neurony
- Θ: threshold obvykle pod jménem bias
- $S(x)$: aktivační funkce
- Y: výstup neuronu

Proces, při kterém se vstupní data postupně předávají z jednoho neuronu na druhý až k výstupu, se nazývá dopředná propagace (forward propagation).

Při dopředné propagaci se vstupní data předají do prvního neuronu, který provede výpočet pomocí aktivační funkce a váh a výsledek předá jako výstup druhému neuronu. Tento proces se opakuje, dokud výstup nedorazí ke konečnému neuronu nebo výstupní vrstvě neuronové sítě.

3.1 Učení neuronové sítě

Cílem neuronové sítě je optimalizovat své váhy a biasy tak, aby dosahovala co nejlepších výsledků na základě trénovacích dat. Trénovací data jsou data, která slouží k učení a nastavení parametrů sítě tak, aby byla schopna správně predikovat výstupy pro nová, dosud neviděná data.

Proces trénování neuronové sítě obecně probíhá ve třech fázích:

Dopředná propagace (Forward propagation): Vstupní data jsou předána sítí, která je pomocí váh a biasů upravuje a předává z neuronu na neuron až k výstupní vrstvě. V této fázi se vypočítávají aktivační funkce a výstupy jednotlivých neuronů.

Zpětná propagace (Backward propagation): Po provedení dopředné propagace se porovnávají výstupy sítě s očekávanými výstupy na základě trénovacích dat. Mezi těmito výstupy je propagována zpět skrze síť a jsou vypočítány gradienty, které udávají směr a míru, pomocí jaké je potřeba upravit váhy a biasy jednotlivých neuronů.

Aktualizace váh: Na základě gradientů zpětné propagace jsou aktualizovány váhy a biasy neuronů, aby byly přizpůsobeny tak, aby minimalizovaly chybu mezi výstupy sítě a očekávanými výstupy. Tím se zlepšuje schopnost sítě predikovat správné výstupy.

Tento proces trénování se opakuje pro každou trénovací sadu dat, aby se síť co nejlépe přizpůsobila dané úloze. Cílem je dosáhnout co nejnižší chyby na trénovacích datech a zároveň dosáhnout

schopnosti generalizovat a správně predikovat výstupy i pro nová, dosud neviděná data (testovací či provozní data).

3.1.1 Učení pod dohledem

Při trénování pod dohledem (supervised) neuronové sítě jsou použita trénovací data jako vzor, na základě kterého je pomocí aktuálních nastavení vah a biasů vypočten výstup sítě viz. dopředná propagace. Tento výstup je následně porovnán s požadovaným výstupem a vypočtena chyba, která udává rozdíl mezi predikovaným a očekávaným výstupem viz. zpětná propagace.

3.1.2 Volné učení

Volné učení (unsupervised) je typ učení, kde trénovací data nejsou doprovázena předem danými a chtěnými výstupy. Na rozdíl od učení pod dohledem (supervised), kde jsou k dispozici páry vstupů s požadovanými výstupy, učení bez dohledu se snaží najít vzorce, struktury nebo skryté informace ve vstupních datech samotných.

Při učení bez dohledu je modelu předloženo pouze množství neoznačených trénovacích dat. Cílem je objevit struktury, shluky, vzory nebo jiné charakteristiky v datech, aniž by byly předem specifikovány. Model se učí tím, že přizpůsobuje své váhy a biasy tak, aby optimalizoval určité cílové kriterium, jako je minimalizace chyby nebo maximalizace pravděpodobnosti dat.

3.2 Rekurentní neuronové sítě

Rekurentní neuronové sítě (Recurrent neural network - RNN) [11] jsou architekturou neuronových sítí, která je navržena speciálně pro práci se sekvenčními daty. Sekvenčními daty se rozumí data, která mají určitý časový nebo prostorový kontext a jsou uspořádána ve formě sekvence nebo posloupnosti.

Hlavním rysem RNN je schopnost uchovávat vnitřní stav (hidden state) a předávat ho ze vstupní vrstvy do následujících časových kroků. To znamená, že RNN může zpracovávat a modelovat závislosti a vztahy mezi jednotlivými prvky v sekvenci.

V kontextu sekvencí je RNN schopna zohlednit předchozí vstupy a stav sítě a použít je k výpočtu aktuálního výstupu. Tím je RNN efektivní pro úkoly, jako je předpovídání dalšího prvku v sekvenci (např. predikce dalšího slova ve větě), překlad sekvenčních dat (jako je strojový překlad) nebo analýza sentimentu v textu.

3.2.1 Zjednodušeně popsané fungování rekurentní neuronové sítě

Rekurentní neuronová síť (RNN) pracuje se sekvencemi dat. V případě věty ‘Já jsem člověk’ by se postupně vkládala jednotlivá slova do sítě a RNN by vytvářela výstupy na základě předchozích informací.

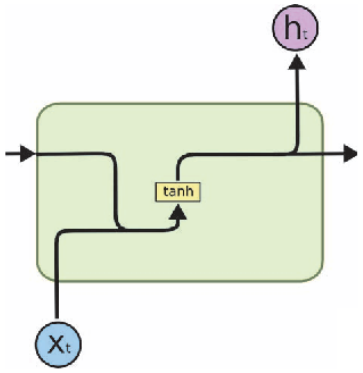
Na začátku sekvence je slovo ‘Já’, které se vloží do sítě, a RNN generuje odpovídající výstup. Poté se slovo ‘jsem’ spolu s informací z předchozího kroku (výstupem zpracování slova ‘Já’) vkládá do sítě a RNN generuje nový výstup. Tímto způsobem se postupně zpracovávají všechna slova v sekvenci.

Důležité je, že RNN udržuje vnitřní stav, který slouží k předávání informace mezi jednotlivými časovými kroky. Při zpracování slova ‘jsem’ se do vstupu RNN přidá i informace z předchozího kroku, která obsahuje informaci o slově ‘Já’. Tímto způsobem se RNN postupně ‘paměťově’ obohacuje o předchozí informace.

Na konci sekvence má RNN teoreticky informace o všech předchozích slovech, protože se vnitřní stav přenáší z jednoho kroku na druhý. Tato schopnost zachytit kontext a závislosti mezi slovy v sekvenci je předností RNN a umožňuje tím efektivně pracovat se sekvenčními daty.

3.2.2 Architektura rekurentních neuronových sítí

Architektura rekurentní neuronové sítě 3.2 jednoho neuronu vypadá následovně.



Obrázek 3.2: RNN architektura [12]

Neuron v rekurentní neuronové síti (RNN) sdílí některé vlastnosti s neurony v běžné neuronové síti. Do neuronu RNN přicházejí vstupní data ze současného kroku (aktuální vstup) a také data z předchozích kroků (vnitřní stav).

Před zpracováním těchto dat se nad nimi provádí aktivační funkce. Jednou z běžně používaných aktivačních funkcí v RNN je hyperbolický tangens. Aktivační funkce hyperbolický tangens převádí vstupní hodnoty na rozmezí mezi -1 a 1.

Aktivační funkce hyperbolický tangens, v následujícím vzorci, se často používá v RNN, protože umožňuje zachování informací z předchozích kroků a přenášení těchto informací do dalších výpočtů. Díky tomu může RNN efektivně zachycovat závislosti na časové ose a pracovat se sekvencemi dat.

$$h_t = \tanh(W_{hh}h_{t-1} + W_{xh}x_t)$$

3.2.2.1 Neduhy rekurentních neuronových sítí

Rekurentní neuronové sítě (RNN) mají problém s dlouhodobými závislostmi a mizejícím gradientem, což omezuje jejich schopnost efektivně zachycovat dlouhodobé informace v sekvencích dat.

Mizející gradient se projevuje tím, že při zpětném šíření chyby se gradient postupně zmenšuje a při trénování se váhy příliš neaktualizují. To znamená, že dlouhodobé závislosti mezi vstupy v sekvenci se obtížněji přenášejí a RNN nemusí být schopná se naučit efektivně využívat informace z dávnějších kroků.

Long Short-Term Memory (LSTM) a Gated Recurrent Unit (GRU) jsou vylepšené architektury RNN, které byly navrženy právě s cílem řešit tyto problémy. Tyto modely používají brány, pro kontrolu probíhajících informací v síti.

LSTM obsahuje speciální paměťovou jednotku, která umožňuje uchovávat a aktualizovat informace na základě brán. Brány umožňují kontrolovat, které informace jsou považovány za důležité a udržovat je v paměti po delší dobu.

Vylepšení GRU je podobné LSTM, ale s jednodušší architekturou. Obsahuje aktualizační bránu, která rozhoduje, jak moc má být aktualizovaná paměťová jednotka, a resetovací bránu, která rozhoduje, jakou část paměti zapomenout. Tímto způsobem se GRU snaží efektivně uchovávat a aktualizovat informace o kontextu.

Tyto vylepšené architektury RNN, jako LSTM a GRU, přinášejí zlepšení v schopnosti zachytit dlouhodobé závislosti a lépe uchovávat kontext při práci se sekvencemi dat.

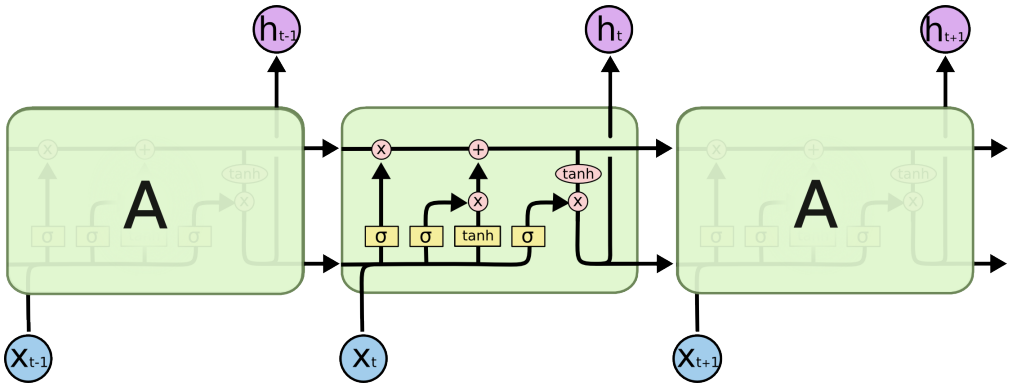
3.3 Long Short-Term Memory

Long Short-Term Memory (LSTM) [13] je speciální verzí rekurentní neuronové sítě, která byla představena v roce 1997 Hochreiterem a Schmidhuberem.

LSTM bylo navrženo specificky k řešení problému s dlouhodobou závislostí a pamatováním informací na delší časový úsek. Jedním z hlavních problémů standardních rekurentních neuronových sítí je, že mají tendenci rychle zapomenout informace z minulých kroků, což omezuje jejich schopnost zachytit dlouhodobé závislosti v datech.

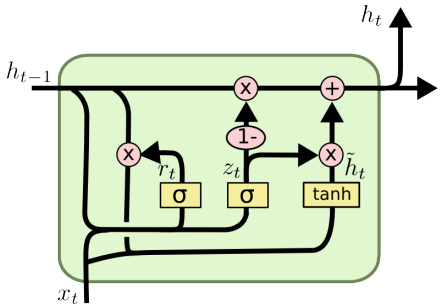
LSTM využívá speciální strukturu paměťové jednotky, která je schopná efektivně uchovávat a aktualizovat informace na základě brán. Tato struktura umožňuje paměťové jednotce rozhodovat, které informace mají být zapomenuty a které mají být uchovány na delší dobu.

Díky svému návrhu LSTM 3.3 umožňuje neuronové síti uchovávat informace o kontextu na delší časový úsek a lépe pracovat s dlouhými sekvencemi dat. To přispívá k větší efektivitě a schopnosti modelu zachytit vztahy a závislosti v datech na delších časových škálách.



Obrázek 3.3: LSTM architektura [13]

Architektura LSTM je oproti standardní RNN značně komplexnější. Hlavním rozšířením jsou právě brány 3.4, které jsou klíčovým prvkem v této architektuře.



$$z_t = \sigma\left(W_z \cdot [h_{t-1}, x_t]\right)$$
$$r_t = \sigma\left(W_r \cdot [h_{t-1}, x_t]\right)$$
$$\tilde{h}_t = \tanh\left(W \cdot [r_t * h_{t-1}, x_t]\right)$$
$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

Obrázek 3.4: LSTM architektura neuronu [13]

LSTM obsahuje tři základní brány: zapomnětlivou bránu (forget gate), vstupní bránu (input gate) a výstupní bránu (output gate). Tyto brány spolu s paměťovou jednotkou (memory cell) umožňují LSTM učit se a efektivně pracovat s dlouhodobými závislostmi v datech. Brány umožňují paměťové jednotce filtrovat a rozhodovat, které informace mají být zapomenuty, které nové informace mají být přidány a které informace mají být využity při výstupu.

3.3.1 Zapomnětlivá brána

Zapomnětlivá brána (forget gate) rozhoduje, jakou část minulého stavu by měl LSTM zapomenout. Na základě vstupů a předchozího stavu určuje, které informace mají být zahozeny.

3.3.2 Vstupní brána

Vstupní brána (input gate) rozhoduje, které nové informace by měly být uloženy v paměťové jednotce LSTM. Pomocí aktivační funkce, využívající sigmoid funkci, určuje, které hodnoty mají být aktualizovány.

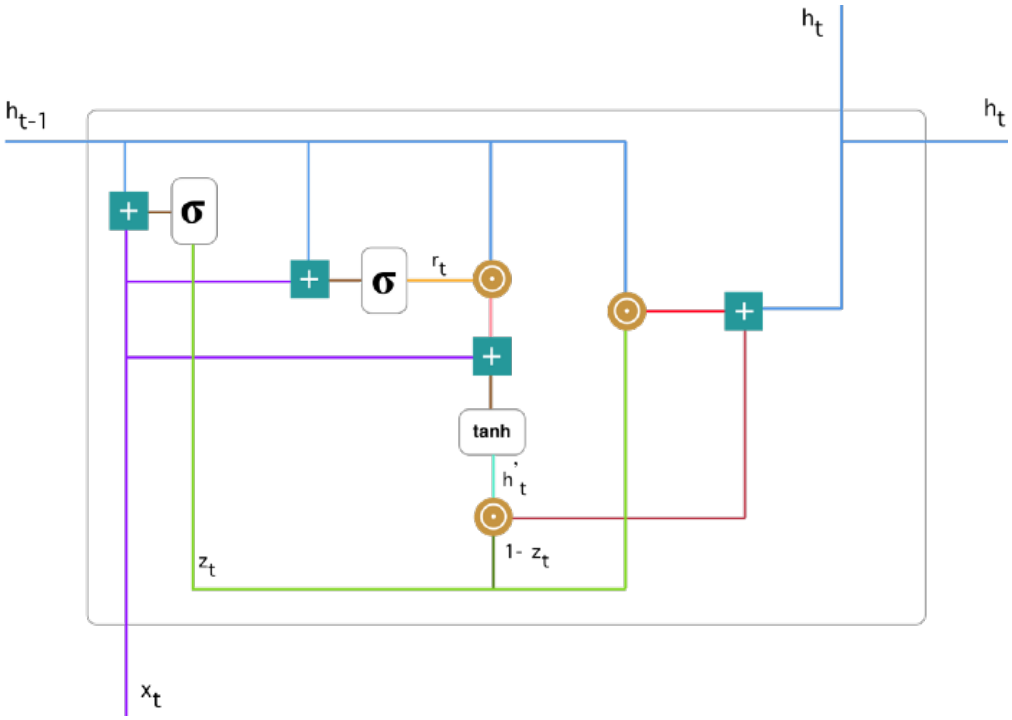
3.3.3 Výstupní brána

Výstupní brána (output gate) rozhoduje, jaká část paměťového stavu by měla být odeslána jako výstup z LSTM. Sigmoidová aktivační funkce ovlivňuje, které hodnoty budou výstupem.

3.4 Gated Recurrent Unit

Gated Recurrent Unit (GRU) [14] je další speciální verze rekurentní neuronové sítě, která byla představena v roce 2014 Cho et al. Architektura GRU je navržena k řešení problému mizejícího gradientu, který je běžným problémem standardních RNN.

Podobně jako LSTM využívá GRU aktualizační a resetovací brány. Tyto brány 3.5 3.6 umožňují GRU učit se zachovávat informace z dřívějších kroků. GRU se snaží efektivně zpracovávat dlouhodobé závislosti a současně snižovat problém mizejícího gradientu.



Obrázek 3.5: GRU architektura neuronu [14]

Obrázek 3.6: GRU význam operací [14]

3.4.1 Aktualizační brána

Aktualizační brána (update gate) rozhoduje, jakou část minulého stavu by měla být aktualizována a která část by měla zůstat nezměněna. Brána pomocí sigmoidové aktivační funkce určuje, jakou část informací z minulého stavu si GRU ponechá a jakou část upraví.

$$z_t = \sigma(W^{(z)}x_t + U^{(z)}h_{t-1})$$

3.4.2 Resetovací brána

Resetovací brána (reset gate) rozhoduje, jakým způsobem se budou kombinovat vstupní data s předchozím stavem. Opět pomocí sigmoidové aktivační funkce určuje, které informace se mají resetovat a které se mají zapomenout.

$$r_t = \sigma(W^{(r)}x_t + U^{(r)}h_{t-1})$$

3.4.3 Porovnání

Vzhledem k popisu problému, jak rozhodnout, zda je text přeložen strojem nebo napsán člověkem, je vhodné využít RNN, konkrétně LSTM a GRU, které jsou navrženy pro zpracování sekvencí dat a řeší problémy původních RNN.

LSTM a GRU jsou obě vylepšené architektury RNN, které se snaží překonat problém mizejícího gradientu a zachovat dlouhodobé závislosti v sekvencích. Oba modely mají některé podobné prvky a často dosahují podobných výsledků. Obě architektury se liší v počtu a funkcionalitě brán a každá má své výhody.

LSTM je považováno za komplexnější a má větší počet parametrů k učení. Je navrženo tak, aby si pamatovalo informace z dlouhodobých závislostí a je schopno lépe zachovávat kontext v delších sekvencích.

Na druhou stranu GRU je jednodušší a má menší počet parametrů. Architektura je navržena tak, aby byla rychlejší a efektivnější v učení. Je vhodnější pro scénáře, kde je omezený počet trénovacích dat nebo kde je důležitá výpočetní efektivita.

Kapitola 4

Popis sítě Transformer

Transformer byl poprvé představen výzkumným týmem z Google Brain [15] v roce 2017 v článku s názvem 'Attention Is All You Need' (Pozornost je vše, co potřebujete) [16]. Tento článek představil novou architekturu založenou na mechanismu self-attention, který se ukázal jako velmi efektivní při zpracování sekvencí a dosáhl vynikajících výsledků v oblasti strojového překladu.

Transformer [17] [18] je architektura hluboké neuronové sítě, která se zaměřuje na problémy sekvence-na-sekvenci, jako je strojový překlad, generování textu nebo analýza sentimentu. Byl vyvinut jako alternativa k rekurentním neuronovým sítím, jako je LSTM a GRU, a přinesl s sebou zásadní inovaci v oblasti zpracování přirozeného jazyka.

Hlavním stavebním prvkem Transformer sítí je self-attention mechanismus, který umožňuje modelu přidělovat váhu různým slovům v sekvenci na základě jejich vzájemného vztahu a kontextu. Tímto způsobem si Transformer dokáže lépe poradit s dlouhodobými závislostmi v sekvencích a zachovat informace o kontextu.

Transformer se skládá z několika vrstev sítě obsahujících self-attention mechanismus a další bloky, jako jsou dopředně šířící se sítě (feed-forward) a normalizační vrstvy. Model je trénován pomocí mechanismu nazvaného 'masked multi-head attention' a optimalizován na základě ztrátové funkce, jako je například cross-entropy.

4.1 Vlastnosti

Transformer je navržen pro zpracování sekvenčních dat, včetně vět v přirozeném jazyce. Jedním z hlavních rozdílů mezi Transformer sítěmi a RNN je právě schopnost Transformer sítí zpracovávat veškerá vstupní data najednou, zatímco RNN musí pracovat sekvenci po sekvenci. Díky mechanismu self-attention dokáže Transformer přiřadit kontext a důležitost jednotlivým slovům v sekvenci, což umožňuje zpracování celých vět najednou.

Díky této vlastnosti má Transformer výhodu v paralelizaci, protože může zpracovávat vstupní data paralelně na větší části modelu. To vede k výraznému zrychlení trénování modelu bez ztráty výkonu. Oproti tomu u RNN je paralelizace složitější kvůli nutnosti zpracovávat data postupně.

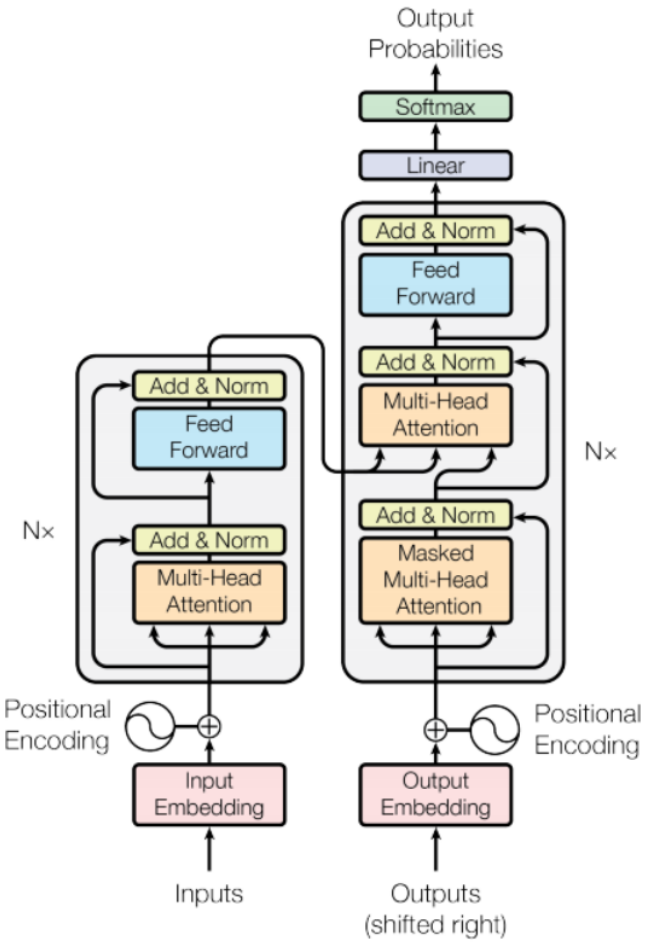
Transformer také využívá techniky pozičního kódování (position encoding) pro zachycení kontextu a pozice jednotlivých slov ve vstupní sekvenci. Pomocí mechanismu attention dokáže mapovat vztahy mezi slovy a lépe porozumět významu a vztahům ve větě.

Před-trénované modely, jako například BERT nebo GPT, jsou vytrénované na velkých datových souborech a poté mohou být doučeny (fine-tuned) na specifické úlohy. Tyto modely představují významný pokrok v oblasti zpracování přirozeného jazyka, protože mají již naučené znalosti o kontextu a vztazích mezi slovy, což usnadňuje jejich aplikaci na různé úlohy a vede ke zlepšení výkonu.

4.2 Architektura

Architektura Transformer sítí 4.1 je založena na mechanismu self-attention, který umožňuje efektivně přiřadit kontext jednotlivým slovům (tokenům) v sekvenci. Transformer se skládá z několika klíčových komponent, které jsou kodér (Encoder), dekodér (Decoder), Multi-head Attention a poziční kódování (Positional Encoding).

Transformer je schopen zpracovávat vstupní sekvence najednou, což umožňuje výrazně paralelizovat výpočty a urychlit trénování modelu. Díky kombinaci self-attention, multi-head attention a dopředného šíření použité v neuronové síti je Transformer schopný efektivně zachytit vztahy mezi slovy a generovat kvalitní výstupy pro různé úlohy, včetně strojového překladu, generování textu a dalších úloh zpracování přirozeného jazyka.



Obrázek 4.1: Architektura Transformer sítí [18]

4.2.1 Kodér

Kodér (Encoder) převádí vstupní sekvenci (např. větu) do vektorové reprezentace. Skládá se z několika identických vrstev, zvaných kódovacích vrstev (encoder layers). Každá vrstva obsahuje dvě subkomponenty: multi-head self-attention mechanismus a dopředně šířící se sítě. Self-attention mechanismus umožňuje modelu získat důležité informace o kontextu jednotlivých slov v rámci sekvence. Neuronová síť s použitím dopředného šíření slouží k transformaci a kombinaci informací z self-attention mechanismu.

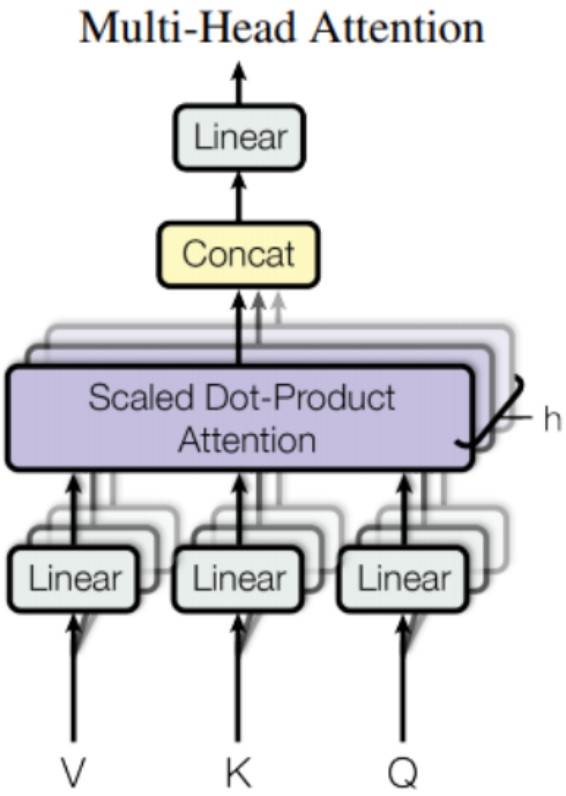
4.2.2 Dekodér

Dekodér (Decoder) generuje výstupní sekvenci (např. přeloženou větu) na základě zakódované vstupní sekvence. Stejně jako kodér, dekodér se skládá z několika identických vrstev, zvaných de-

kódovací vrstvy (decoder layers). Každá vrstva obsahuje tři subkomponenty: masked multi-head self-attention mechanismus, který umožňuje dekodéru vidět pouze předchozí pozice, multi-head attention mechanismus, který se zaměřuje na kódovaný kontext, a neuronovou síť pro transformaci a kombinaci informací.

4.2.3 Multi-head Attention

Attention mechanismus je klíčovou součástí sítí Transformer. Umožňuje modelu přiřadit různou důležitost různým částem vstupní sekvence při generování výstupu. Transformer používá multi-head attention 4.2, což znamená, že attention mechanismus je aplikován několikrát s různými projekcemi vstupních vektorů. To umožňuje modelu získat různé perspektivy na vztahy mezi slovy.



Obrázek 4.2: Multi-head Attention [18]

Každá attention head jednotka využívá tři maticové transformace (Wq, Wk, Wv) pro dotazování (query), klíče (keys) a hodnoty (values) vstupních slov (tokenů).

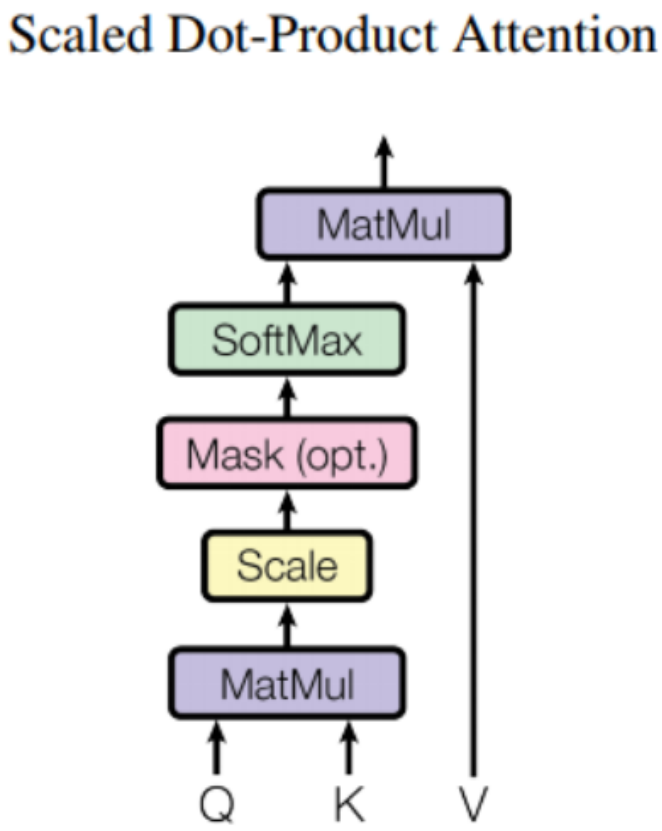
Výpočet attention se provádí paralelně pro všechny attention head jednotky, což umožňuje efektivní zpracování. Výsledky z jednotlivých attention head jednotek jsou následně spojeny a předány dále pomocí neuronové sítě využívající dopředné šíření do dalších vrstev transformace.

Tímto způsobem model může získat různé informace o vztazích a relevanci mezi slovy v rámci vstupní sekvence. Každá attention head jednotka se může zaměřit na specifické aspekty, jako je sledování kontextuálních závislostí, vazeb mezi slovy nebo jiné relevantní informace.

4.2.3.1 Počítání techniky Attention

Transformer využívá scaled dot-product attention 4.3 jako mechanismus pro výpočet vah mezi slovy (tokeny) vstupní sekvence.

Při výpočtu scaled dot-product attention jsou vstupní sekvence transformovány pomocí maticových transformací (Wq, Wk, Wv) na dotazovací (query), klíčové (keys) a hodnotové (values) vektory. Poté se vypočítá podobnost (dot product) mezi dotazovým vektorem a klíčovými vektory.

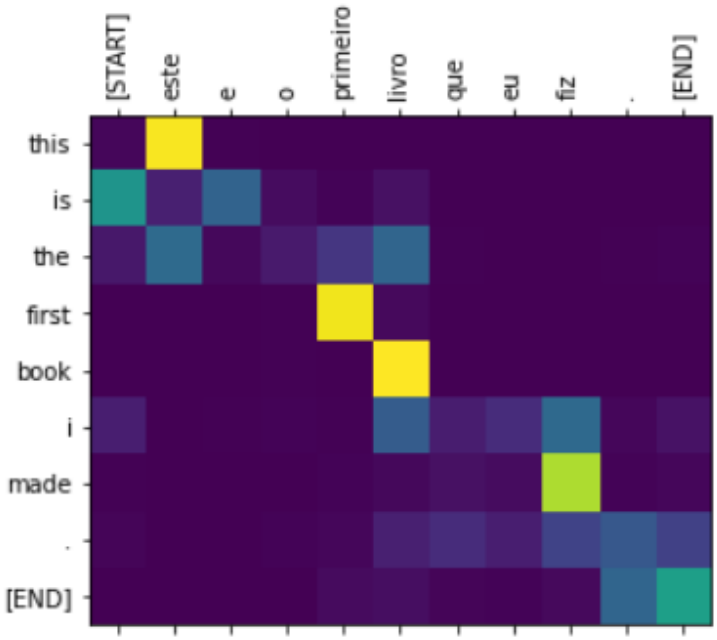


Obrázek 4.3: Scaled Dot-Product Attention [18]

Jedním z klíčových kroků je normalizace podobnosti pomocí škálování, aby se zabránilo problémům s velkými hodnotami např. stabilita gradientu. Následně s využitím Softmax funkce (viz. následující vzorec) se získávají normalizované váhy pro každý token v rámci sekvence, což reprezentuje jeho relevanci vůči ostatním tokenům.

$$Attention(Q, K, V) = softmax(QK^T / \sqrt[2]{d_k})V$$

Následně se váženě kombinují hodnotové vektory (values) s váhami, což vytváří vektor vkládání slov pro každý token v kontextu. Tento vektor reprezentující vkládána slova 4.4 obsahuje informaci o samotném tokenu a jeho vztahu k ostatním relevantním tokenům v sekvenci.



Obrázek 4.4: Prezentována technika Attention [18]

4.2.4 Poziční kódování

Pro zachycení pořadí slov v sekvenci používá Transformer poziční kódování (positional encoding) [19]. Jedná se o speciální vektorovou reprezentaci, která je přidána ke vstupním slovům (tokenům) a poskytuje informaci o jejich pozici v sekvenci. Tímto způsobem se zachovává informace o pořadí slov i při zpracování vstupu bez rekurentního propojení.

4.2.4.1 Vstupní data

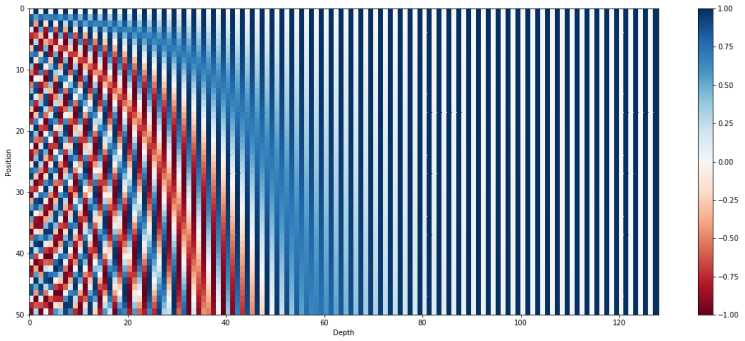
Ve vstupním procesu Transformer sítí jsou slova reprezentována pomocí techniky vkládání slov, která je často založena na před trénovaných slovních vektorech. Tyto vektory mají pevnou dimenzi, která se obvykle pohybuje v rozmezí desítek až stovek.

Technika vkládání slov zachycuje významové vztahy mezi slovy, ale nezahrnuje informaci o jejich pozici v sekvenci. Proto je k zachování informace o pořadí slov v Transformer sítí použita technika nazývaná poziční kódování 4.5 (viz. následující vzorce).

Pozici slov (tokenů) v sekvenci určuje kombinace hodnot vektoru pozičního kódování a vektoru reprezentující vkládána slova. Tyto informace jsou následně vstupem do kodéru a dekodéru Transformer sítí, který na základě nich provádí operace techniky attention a generuje výstupní sekvenci.

$$PE_{(pos,2i)} = \sin(pos/10000^{2i/d_{model}})$$

$$PE_{(pos,2i+1)} = \cos(pos/10000^{2i/d_{model}})$$



Obrázek 4.5: Zobrazení Positional Encoding techniky pro 128D vektor [19]

4.3 Trénování modelu

Transformer sítě se často učí pod samo dohledem (self-supervised), který kombinuje volné učení (unsupervised) na velkém datovém souboru a následně doučeny (fine-tuned) pomocí učení pod dohledem (supervised).

Při volném učení je Transformer modelu prezentováno velké množství neoznačených dat, například textové korpusy získané z Internetu, které jsou bohaté na přirozený jazyk. Model se učí predikovat následující slovo v sekvenci, rekonstruovat chybějící části vstupní sekvence nebo provádět jiné úlohy, které vyžadují porozumění kontextu a vztahům mezi slovy. Tímto učením se model naučí efektivně zpracovávat textová data, zachytávat jazykové vzorce a získávat všeobecné znalosti o vztazích mezi slovy.

Po volném učení trénování je model doučen pomocí dohledového učení na menším označeném datasetu specifickým pro konkrétní úlohu. Například pro sentimentální analýzu se používá dataset obsahující texty s označeným sentimentem, pro modelování jazyka se používá textový dataset s označenými sekvencemi.

Tímto kombinovaným přístupem se dosahuje dobrého výkonu na konkrétních úlohách zpracování přirozeného jazyka, jako je sentimentální analýza, generování následujících vět, modelování jazyka nebo odpovídání na otázky. Transformer sítě díky své schopnosti zachycovat kontext a vztahy mezi slovy dosahují v těchto úlohách často nejlepšího (state-of-the-art) výkonu.

4.4 Aplikování

Transformer sítě se ukázaly jako velmi úspěšné v oblasti zpracování přirozeného jazyka a na mnoha úlohách, včetně strojového překladu a časové závislých (time series) předpovědí. Jejich schopnost zachycovat kontext a vztahy mezi slovy a efektivně zpracovávat sekvence je velmi výhodná pro tyto úlohy.

Nicméně, Transformer sítě mají široké možnosti využití i v jiných oblastech. Před-trénované modely, jako je GPT-2, se ukázaly jako úspěšné v generování textu, ať už jde o tvorbu nových článků, dialogů, nebo dokonce celých dokumentů. Díky schopnosti zachycovat kontext a syntaktické struktury vět dokáží tyto modely produkovat texty s vysokou kvalitou a přirozeností.

Transformer sítě se také osvědčily v oblasti analýzy biologických sekvencí, kde jsou schopny zpracovávat sekvence DNA nebo proteinů a hledat vzorce a vztahy mezi nimi. V oblasti porozumění videím se Transformer sítě využívají k analýze videí a extrakci informací z video sekvencí.

Dalším zajímavým příkladem je využití Transformer sítí v šachových algoritmech, kde mohou být před-trénované modely, jako je GPT-2, doučeny na hrání šachu. Tyto modely se naučí strategie a pravidla hry a mohou dosahovat vysoké úrovně ve hře proti lidským hráčům.

Transformer sítě tedy nabízejí široké spektrum využití a mohou být adaptovány na různé oblasti, kde je důležité zpracovávat sekvence dat a zachycovat jejich vztahy a strukturu.

4.5 BERT

BERT (Bidirectional Encoder Representations from Transformers) [20] 4.6 je architektura založená na Transformer modelu pro zpracování přirozeného jazyka (NLP). Byl vytvořen týmem Google Brain [15] a poprvé publikován v roce 2018.

BERT přinesl významný pokrok v oblasti NLP a dosáhl několika nejlepších (state-of-the-art) výsledků v různých NLP úlohách. Jeho klíčovou vlastností je schopnost zachytit vztahy mezi slovy a jejich kontextem v obou směrech, tedy v předcházejícím i následujícím kontextu. Tímto způsobem BERT vytváří velmi bohaté a reprezentativní vektorové reprezentace slov a vět.

Díky své schopnosti zachytit jazykový kontext se BERT stal populárním a v roce 2020 byl hojně využíván ve strojovém překladu z anglického jazyka do jiných jazyků. Jeho před-trénované váhy a modelové znalosti umožňují efektivní adaptaci na různé úlohy pomocí doučující techniky.

BERT otevřel cestu pro další vylepšené architektury v oblasti NLP. Tato architektura se stala důležitým nástrojem pro různé úlohy zpracování přirozeného jazyka, včetně strojového překladu, rozpoznávání entit, sentiment analýzy a dalších.

Původně veřejně publikován BERT má dva typy konfigurace pro sdílení.

- BERT BASE, který obsahuje 12 enkodéru s 12ti self-attention head jednotkami
- BERT LARGE s 24 enkodéry a 16ti self-attention head jednotkami

BERT je přetrénován na neoznačených (unsupervised) jazykových reprezentacích, což znamená, že trénovací data, na kterých je trénován, nejsou specificky označena pro určitý výstup. Místo toho jsou použity rozsáhlé textové korpusy, které byly získány ze stránek jako jsou např. Wikipedie a BooksCorpus, které obsahují obecný textový materiál z různých zdrojů.

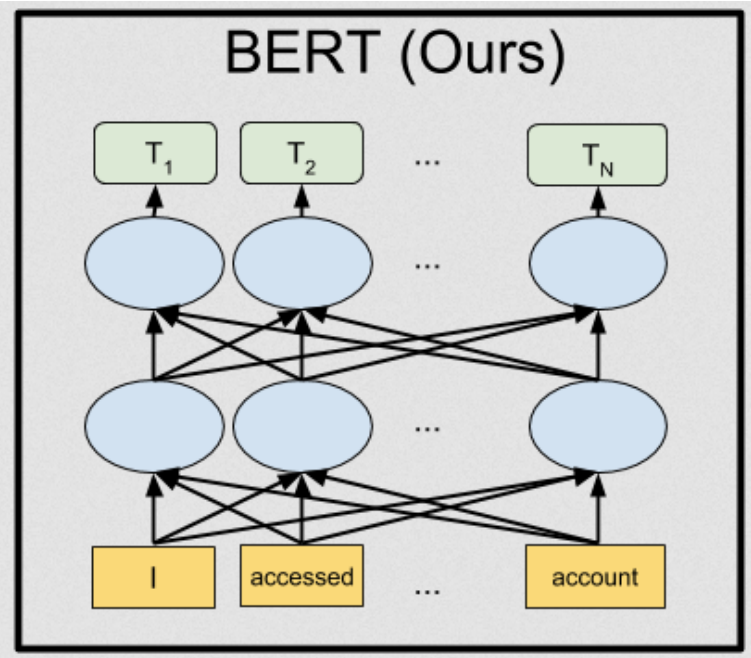
Díky před trénování na rozsáhlých a různorodých textových korpusů BERT získává obecné jazykové znalosti, které mu umožňují lépe porozumět a reprezentovat různé textové vstupy. Tato schopnost představuje výhodu při aplikacích NLP, kde je modelu potřeba porozumět a zachytit různé jazykové nuance a vztahy.

BERT bere v potaz kontext slova, což mu umožňuje rozlišovat význam slova v závislosti na jeho kontextu. Rozdíl ve významu slova 'koruna' v následujících větách by měl být reflektován v reprezentaci chování BERT modelu.

- Česká koruna se propadla nejvíce v historii ČR.
- V korunách stromů sedí ptáci.
- Král nosí korunu.

BERT je obousměrným (bidirectional) modelem, což znamená, že bere v potaz kontext slova z obou směrů. Tato vlastnost je dosažena pomocí mechanismu nazývaného maskované modelování jazyka (masked language modeling), kdy BERT se učí předpovídat chybějící slova v textu. Při trénování jsou náhodně vybraná slova věty maskována a model se učí předpovídat původní tvar těchto slov na základě kontextu, který zahrnuje jak slova před nimi, tak slova za nimi.

Díky obousměrnému zohledňování kontextu je BERT schopen zachytit slovní vztahy a významy ve větách, které jsou důležité pro porozumění celému textu. Tím se odstraňuje omezení jednosměrných (unidirectional) modelů, které sledují kontext pouze z jednoho směru. Obousměrný přístup BERTa umožňuje lépe modelovat závislosti mezi slovy a vytvářet kontextuální reprezentace jednotlivých slov na základě celé věty.



Obrázek 4.6: BERT model [21]

4.5.1 Ukázka obousměrného přístupu

V následující větě ‘Já jsem Honza’ BERT s obousměrným přístupem je schopen zachytit vztah slova ‘jsem’ nejen k předchozímu slovu ‘Já’, ale také k následujícímu slovu ‘Honza’. Díky své schopnosti zohledňovat kontext z obou směrů dokáže BERT lépe porozumět vztahům mezi slovy ve větě.

V případě jednosměrného modelu by pouze kontext předcházející slovu ‘jsem’ nebo kontext následující slova ‘jsem’ byl brán v úvahu při modelování vztahu s tímto slovem. To by mohlo vést k omezenému porozumění celému kontextu věty.

Díky schopnosti BERTa zohledňovat kontext ze všech stran je schopen zachytit významy obou slov ‘Já’ a ‘Honza’ a lépe reprezentovat jejich vzájemný vztah ve větě ‘Já jsem Honza’. Tímto způsobem BERT dokáže lépe porozumět významům jednotlivých slov a jejich vztahům v celém kontextu věty.

Jednosměrný přístup, který je běžně používán v tradičních jazykových modelech, neumožňuje předvídat budoucí slova na základě slov, která jej předcházela a následovala. Tím pádem by při jednosměrném modelování nebylo možné získat úplný kontext věty.

BERT řeší tento problém pomocí techniky maskování (masking) 4.7. Při trénování BERTa jsou náhodně vybraná slova vstupního textu maskována a následně je model trénován na předpovídání těchto zakrytých slov na základě okolního kontextu. Tímto způsobem BERT získává schopnost porozumět kontextu celé věty a naučit se předpovídat nejen slova na základě předchozích, ale i na základě budoucích slov, která jsou v daném kontextu maskována.

Input: The man went to the [MASK]$_1$. He bought a [MASK]$_2$ of milk .
Labels: [MASK]$_1$ = store; [MASK]$_2$ = gallon

Obrázek 4.7: BERT případ maskování [21]

Technika maskování slov ve vstupním textu a následná predikce těchto slov je známá už delší dobu. Co je unikátní u BERTa, je jeho schopnost efektivně využívat tuto techniku v kombinaci s obousměrným kontextovým modelováním.

Co se týče porozumění vztahům mezi větami, BERT má v sobě zakódovanou informaci o vzájemném vztahu dvou vět v tzv. segmentovém vektoru (segment embedding vector). Segmentový vektor 4.8 je část reprezentace vstupního textu, která specifikuje, které slova a věty patří do kterých segmentů. BERT je tak schopen rozpoznat, zda věta 'A' následuje po větě 'B' nebo naopak.

Sentence A = The man went to the store.
Sentence B = He bought a gallon of milk.
Label = IsNextSentence

Sentence A = The man went to the store.
Sentence B = Penguins are flightless.
Label = NotNextSentence

Obrázek 4.8: BERT příklad sekvence [21]

4.5.2 Využití

Jedna z výhod Transformer sítí je možnost využití před-trénovaných modelů pro různé úlohy zpracování přirozeného jazyka.

Před-trénované modely jako je BERT jsou natrénovány na obrovských množstvích textových dat a jsou schopny zachytit bohaté vztahy mezi slovy a kontextem. Tyto modely jsou takovým druhem jazykového modelu, který vytváří vektorové reprezentace slov a vět na základě jejich kontextu.

Pomocí techniky doučení je možné před-trénovaný model BERT přizpůsobit na specifický úkol, například pro rozpoznávání sentimentu, strojový překlad nebo jiné úlohy zpracování přirozeného jazyka. Při doučování se upravují pouze některé části modelu, zatímco váhy a znalosti z před trénované fáze zůstávají zachovány.

Kapitola 5

Detekce strojově přeložených textů

Pro praktickou ukázku byl celý kód napsán v programovacím jazyce Python [19] s hlavním využitím knihovny Tensorflow. Jako prostředí pro testování byl zvolen Google Colab [22], ale pro rozsáhlejší testování byl využit školní server VŠB.

Pro tuto praktickou ukázku byly vybrány dva různé datové soubory. První soubor obsahuje pouze anglický jazyk, zatímco druhý soubor obsahuje texty ze 24 různých jazyků.

Pro každý z těchto datových souborů bylo vytvořeno několik modelů 5, které byly dále pozorovány a laděny. V následující části budou uvedeny výsledky těchto modelů pro každý datový soubor.

- LSTM model
- GRU model
- LSTM model s technikou GloVe embedding
- GRU model s technikou GloVe embedding
- LSTM model s technikou Attention
- GRU model s technikou Attention
- BERT

5.1 Postup pro vytvoření klasifikátorů

- Příprava trénovacího datasetu: Nejprve je zapotřebí mít k dispozici trénovací dataset, který obsahuje textová data a odpovídající klasifikační štítky. Dataset by měl být rozdělen na trénovací, validační a testovací část.

- Předzpracování dat: Textová data musí být převedena do vhodného formátu pro zpracování strojem. To zahrnuje tokenizaci textu.

- Vytvoření vektorové reprezentace: Textová data musí být reprezentována pomocí vektorů, které jsou zpracovatelné strojem. K tomu se často využívá technika vkládání slov (word embeddings), která převádí tokeny na vektorovou reprezentaci. Využije se čistě vytvořená reprezentace vkládání slov z trénovacího datasetu a GloVe.

- Návrh architektury klasifikátoru: Je třeba vybrat vhodnou architekturu klasifikátoru, která bude použita pro trénování. Mezi běžné architektury pro klasifikaci textu patří rekurentní neuronové sítě (např. LSTM, GRU) nebo transformer modely (jako BERT).

- Trénování klasifikátoru: Architektura klasifikátoru je trénována na trénovacím datasetu pomocí metody zpětného šíření chyby (backpropagation). Během trénování jsou optimalizovány váhy a biasy modelu tak, aby minimalizovaly chybu při predikci klasifikačních štítků. Trénování může probíhat po několik epoch, přičemž v každé epoše jsou data prezentována modelu v různém pořadí (pro lepší generalizaci).

- Optimalizace a ladění modelu: Po dokončení trénování je třeba model optimalizovat a ladit. To může zahrnovat úpravu hyperparametrů modelu (např. learning rate, velikost skrytých vrstev, počet vrstev), aplikaci regularizace (např. dropout, Gaussovský dropout) a další techniky pro zvýšení výkonnosti a generalizace modelu.

- Evaluace modelu: Model je vyhodnocen na validačním a testovacím datasetu pomocí metriky přesnosti. Na základě výsledků je možné porovnávat různé klasifikátory a vybrat ten nejlepší.

- Inferenční fáze: Po úspěšném trénování a vyhodnocení je model připraven k použití pro predikce na nových neznámých datech. Je možné jej aplikovat na reálné problémy a získat predikce nebo klasifikace na základě nových vstupů.

Celý proces vytváření klasifikátorů je iterativní a zahrnuje opakované testování, ladění a optimalizaci modelu s cílem dosáhnout co nejlepších výsledků.

5.1.1 Datové soubory

Datové soubory byly vytvořeny autorem této práce. Na počátku bylo navrženo získání odstavců napsaných lidmi na radu vedoucího práce. Tato data byla získána z webových stránek Evropské unie [23], které obsahují texty ve všech jazycích, do kterých autoři a překladatelé přispívají.

Po očištění tohoto textového korpusu byla proveden strojový překlad pomocí několika nástrojů, jako je Deepl [24] a Google Translate [25].

Aby byl zachován rozdíl mezi reálnými daty a přeloženými daty, bylo stanoveno důležité pravidlo. Každý odstavec napsaný lidmi byl přeložen pouze do jednoho jazyka, aby byl datový soubor rovnoměrně rozdělen mezi reálná a přeložená data.

5.1.2 Dataset pouze s anglickými texty

Tento dataset obsahuje pouze texty psané v anglickém jazyce, které budou sloužit k trénování klasifikátorů pro provedení detekce.

V následující tabulce 5.1 jsou uvedeny informace o datovém souboru obsahujícím pouze anglicky psané texty.

Zmenšená velikost přeložených odstavců je způsobena rozdíly v počtu původních odstavců v různých jazycích. Každý jazyk se liší a také spisovatelé či překladatelé přispívají ke specifičnosti každého jazyka.

Tabulka 5.1: Rozložení datového souboru pro Angličtinu

Typ	Počet
Reálné odstavce	3398
Přeložené odstavce	3437

5.1.3 Dataset přeložen do všech jazyků

Tento datový soubor obsahuje texty napsané ve všech jazycích 5.1.3, které se vyskytují na stránkách Evropské unie.

Celkový počet jazyků v tomto datovém souboru je 24.

- Bulgarian, Croatian, Czech, Danish, Dutch, English,
- Estonian, Finland, French, German, Greek, Hungarian,
- Irish, Italian, Latvian, Lithuanian, Maltese, Polish,
- Portuguese, Romania, Slovakia, Slovene, Spanish, Sweden

V následující tabulce 5.2 jsou uvedeny informace o datovém souboru obsahujícím pouze anglicky psané texty.

Tabulka 5.2: Rozložení více jazyčného datového souboru

Typ	Počet
Reálné odstavce	38528
Přeložené odstavce	38581

5.1.4 Trénování

Klasifikátory byly trénovány na základě svých individuálních parametrů a konfigurací, při použití optimálního počtu epoch pro každý z nich.

Po opakovaném trénování klasifikátorů se projevil jev přetrénování na trénovacích datech. K tomu účelu byly přidány Dropout vrstvy a regularizátory s cílem potlačit tento jev. Nicméně i přes tyto přidané techniky se nepodařilo přetrénování úplně eliminovat. Po důkladném zkoumání chování klasifikátorů, které byly přetrénovány, se ukázalo, že detekce na testovacích datech byla lepší než detekce u klasifikátorů, kterým bylo trénování ukončeno před dosažením přetrénování.

Na základě těchto výsledků bylo provedeno upravení trénování všech modelů. První fáze trénování spočívala v natrénování modelu na omezeném počtu epoch a následně byl vybrán model s nejlepší binární přesností (binary accuracy). Ve druhé fázi trénování byla použita metoda hrubé síly k dalšímu zlepšení modelu. V každé epoše byl model přetrénován a poté byl vybrán ten model, který dosáhl lepší hodnoty validační přesnosti (binary validation accuracy). Tímto přístupem bylo dosaženo postupného zlepšování modelu a výběru nejlepšího možného výsledku.

Díky tomuto dvoucestnému trénování se model nejdříve naučí základní vzorce a struktury z trénovacích dat, což mu umožní získat základní znalosti a porozumění danému problému. Následně, v druhé fázi trénování, dochází k dalšímu zlepšení modelu pomocí metody hrubé síly. Tím se zvyšuje jistota, s jakou klasifikátor provádí predikce.

Dvojí trénování tak umožňuje kombinovat přínosy základního naučeného vzorce s vylepšenými schopnostmi a jistotou predikce.

5.1.5 LSTM a GRU modely

Klasifikátor využívá architekturu LSTM (Long Short-Term Memory) nebo GRU (Gated Recurrent Unit) společně s technikou obousměrných vrstev (Bidirectional) pro efektivní zachycení kontextu jednotlivých tokenů. Pro potlačení přetrénování na trénovacích datech je v klasifikátoru implementována Gaussian-Dropout vrstva. Optimalizace vah klasifikátoru je prováděna pomocí optimalizátoru RMSprop, který se zaměřuje na adaptivní aktualizaci vah s ohledem na historii gradientů. Pro aktivační funkci v klasifikátoru byla zvolena Sigmoid, která transformuje výstupy neuronů na pravděpodobnostní hodnoty mezi 0 a 1.

5.1.5.1 Technika GloVe

K modelu LSTM nebo GRU je připojen Embedding, který je odvozen z již natrénovaného vektorového mapování GloVe. Tímto způsobem bude mít model naučené vztahy mezi slovy, které byly trénovány na rozsáhlejších datových sadách než je samotný trénovací dataset.

K modelu LSTM nebo GRU je přidávána Attention vrstva, která je moderní technikou pro výpočet vah jednotlivých slov v sekvenci. Tímto způsobem má model možnost porozumět významu jednotlivých slov a upřednostnit ta, která jsou významná.

5.1.5.2 Technika Attention

K modelu LSTM nebo GRU je přidávána Attention vrstva, která je moderní technikou pro výpočet vah jednotlivých slov v sekvenci. Tímto způsobem má model možnost porozumět významu jednotlivých slov a upřednostnit ta, která jsou významná.

5.1.6 BERT

Klasifikátor využívá před-trénovaný model BERT, ke kterému je připojena Dropout vrstva a vrstva neuronové sítě s jedním neuronem pro binární detekci.

5.2 Výsledné klasifikátory anglického datasetu

5.2.1 Architektura klasifikátoru LSTM

Samotný model 5.3 se skládá z Embedding vrstvy, který překládá tokeny do 256 dimenzionálního vektoru, následované dvěma obousměrnými LSTM vrstvami, přičemž každá z těchto vrstev obsahuje 128 neuronů. Poté následují čtyři neuronové vrstvy s postupně se snižujícím počtem neuronů: 64, 32, 16 a 1. Mezi každou vrstvou modelu se nachází Dropout vrstvy nebo regularizující vrstvy. Pro optimalizaci vah modelu byl zvolen optimalizátor RMSprop a jako výpočetní metrika byla zvolena binární přesnost. Celkový počet parametrů modelu činí 3 179 905.

Tabulka 5.3: LSTM model

Vrstva modelu	výstupný formát dat	počet parametrů
TextVectorization	(None, 256)	0
Embedding	(None, 256, 256)	2 372 352
Bidirectional	(None, 256, 256)	394 240
Gaussian Noise	(None, 256, 256)	0
Bidirectional	(None, 256)	394 240
Gaussian Noise	(None, 256)	0
Dense	(None, 64)	16 448
Gaussian Noise	(None, 64)	0
Dense	(None, 32)	2 080
Gaussian Noise	(None, 32)	0
Dense	(None, 16)	528
Dropout	(None, 16)	0
Dense	(None, 1)	17

5.2.2 Architektura klasifikátoru GRU

Samotný model 5.4 se skládá z Embedding vrstvy, který překládá tokeny do 256 dimenzionálního vektoru, následované dvěma obousměrnými GRU vrstvami, přičemž každá z těchto vrstev obsahuje 128 neuronů. Poté následují čtyři neuronové vrstvy s postupně se snižujícím počtem neuronů: 64, 32, 16 a 1. Mezi každou vrstvou modelu se nachází Dropout vrstvy nebo regularizující vrstvy. Pro optimalizaci vah modelu byl zvolen optimalizátor RMSprop a jako výpočetní metrika byla zvolena binární přesnost. Celkový počet parametrů modelu činí 2 984 321.

Tabulka 5.4: GRU model

Vrstva modelu	výstupný formát dat	počet parametrů
TextVectorization	(None, 256)	0
Embedding	(None, 256, 256)	2 372 352
Bidirectional	(None, 256, 256)	296 448
Bidirectional	(None, 256)	296 448
Dense	(None, 64)	16 448
Gaussian Noise	(None, 64)	0
Dense	(None, 32)	2 080
Gaussian Noise	(None, 32)	0
Dense	(None, 16)	528
Dense	(None, 1)	17

5.2.3 Architektura klasifikátoru LSTM s technikou GloVe

V LSTM modelu je obecná Embedding vrstva nahrazena GloVe Embedding 5.5 vrstvou viz. 5.1.5.1. Celkový počet parametrů modelu činí 2 604 009.

Tabulka 5.5: LSTM model s technikou GloVe

Vrstva modelu	výstupný formát dat	počet parametrů
Text Vectorization	(None, 256)	0
GloVe	(None, 256, 256)	1 853 800
Bidirectional	(None, 256, 256)	336 896
Gaussian Noise	(None, 256, 256)	0
Bidirectional	(None, 256)	394 240
Gaussian Noise	(None, 256)	0
Dense	(None, 64)	16 448
Gaussian Noise	(None, 64)	0
Dense	(None, 32)	2 080
Gaussian Noise	(None, 32)	0
Dense	(None, 16)	528
Dropout	(None, 16)	0
Dense	(None, 1)	17

5.2.4 Architektura klasifikátoru GRU s technikou GloVe

V GRU modelu je obecná Embedding vrstva nahrazena GloVe Embedding 5.6 vrstvou viz. 5.1.5.1. Celkový počet parametrů modelu činí 2 422 761.

Tabulka 5.6: GRU s technikou GloVe model

Vrstva modelu	výstupný formát dat	počet parametrů
Text Vectorization	(None, 256)	0
GloVe	(None, 256, 256)	1 853 800
Bidirectional	(None, 256, 256)	253 440
Gaussian Noise	(None, 256, 256)	0
Bidirectional	(None, 256)	296 448
Gaussian Noise	(None, 256)	0
Dense	(None, 64)	16 448
Gaussian Noise	(None, 64)	0
Dense	(None, 32)	2 080
Gaussian Noise	(None, 32)	0
Dense	(None, 16)	528
Dropout	(None, 16)	0
Dense	(None, 1)	17

5.2.5 Architektura klasifikátoru LSTM s technikou Attention

Do LSTM modelu 5.7 byla přidána Attention vrstva 5.1.5.2 a došlo k další změně optimalizátoru z RMSprop na Adam. Celkový počet parametrů modelu nyní činí 3 180 417.

Tabulka 5.7: LSTM s technikou Attention model

Vrstva modelu	výstupný formát dat	počet parametrů
Text Vectorization	(None, 256)	0
Embedding	(None, 256, 256)	2 372 352
Bidirectional	(None, 256, 256)	394 240
Gaussian Noise	(None, 256, 256)	0
Bidirectional	(None, 256, 256)	394 240
Gaussian Noise	(None, 256, 256)	0
Attention	(None, 256)	512
Gaussian Noise	(None, 256)	0
Dense	(None, 64)	16 448
Gaussian Noise	(None, 64)	0
Dense	(None, 32)	2 080
Gaussian Noise	(None, 32)	0
Dense	(None, 16)	528
Dropout	(None, 16)	0
Dense	(None, 1)	17

5.2.6 Architektura klasifikátoru GRU s technikou Attention

Do GRU modelu 5.8 byla přidána Attention vrstva 5.1.5.2 a došlo k další změně optimalizátoru z RMSprop na Adam. Celkový počet parametrů modelu nyní činí 2 984 833.

Tabulka 5.8: GRU s technikou Attention model

Vrstva modelu	výstupný formát dat	počet parametrů
Text Vectorization	(None, 256)	0
Embedding	(None, 256, 256)	2 372 352
Bidirectional	(None, 256, 256)	296 448
Gaussian Noise	(None, 256, 256)	0
Bidirectional	(None, 256, 256)	296 448
Gaussian Noise	(None, 256, 256)	0
Attention	(None, 256)	512
Gaussian Noise	(None, 256)	0
Dense	(None, 64)	16 448
Gaussian Noise	(None, 64)	0
Dense	(None, 32)	2 080
Gaussian Noise	(None, 32)	0
Dense	(None, 16)	528
Dropout	(None, 16)	0
Dense	(None, 1)	17

5.2.7 Architektura klasifikátoru BERT

Tento model 5.9 se skládá z několika vrstev: Input vrstvy, preprocessoru pro přetrénovaný model BERT, samotného modelu BERT a přídavné neuronové sítě. První vrstvou je Input vrstva, která slouží k přijímání textových dat. Následuje preprocessor, který transformuje text do formátu vhodného pro BERT. To zahrnuje převod velkých písmen na malá, odstranění interpunkčních znaků a další úpravy. Poté dochází k převodu sekvence slov na sekvenci tokenů a následně na vektory. BERT model zpracovává tokeny a vytváří kontextovou pozornost (contextual attention) mezi jednotlivými tokeny. Vrstva s BERT modelem je sestavena ze 4 skrytých vrstev, kde každá vrstva obsahuje 512 neuronů a 8 attention head jednotek. Pro regulaci a potlačení přetrénování je přidána Dropout vrstva. Nakonec je připojena Dense vrstva s jedním neuronem, která slouží k binární klasifikaci. Pro optimalizaci vah modelu byl zvolen optimalizátor AdamW a jako výpočetní metrika byla zvolena binární přesnost. Celkový počet parametrů modelu činí 28 764 162.

Architektura tohoto modelu je výrazně odlišná od ostatních modelů a očekává se, že BERT bude převládat nad ostatními.

Tabulka 5.9: BERT model

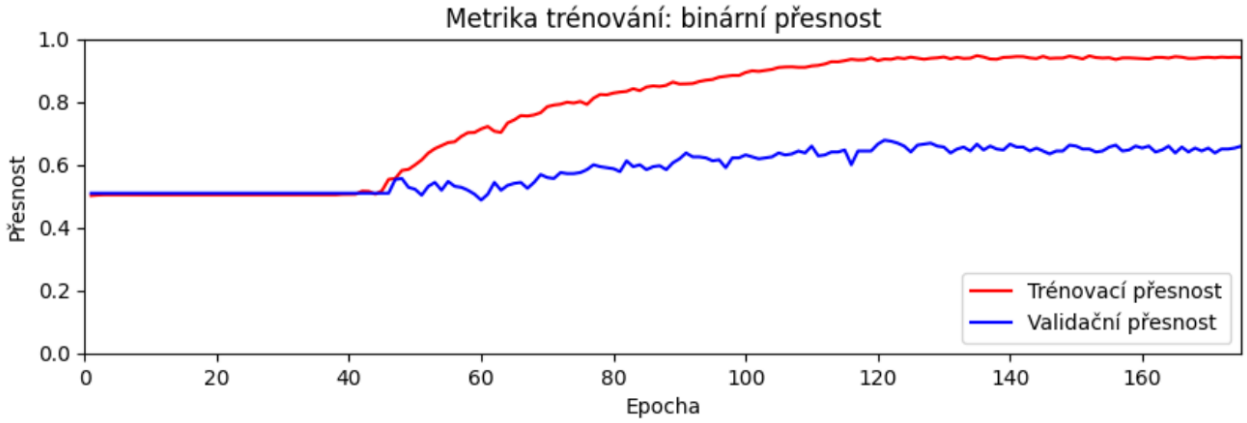
Vrstva modelu	výstupný formát dat	počet parametrů
InputLayer	(None,)	0
Preprocessing	—	0
BERT encoder	—	28,763,649
Dropout	(None, 512)	0
Dense	(None, 1)	513

5.2.8 Výsledky klasifikátorů

5.2.8.1 Trénovací fáze

V následujících grafech 5.1 5.2 5.3 5.4 5.5 5.6 5.7 jsou znázorněny křivky učení v průběhu postupně vykonávaných epoch. Červená křivka představuje přesnost predikce modelu na trénovacích datech a modrá křivka představuje přesnost predikce modelu na validačních datech.

Graf trénování 5.1 LSTM modelu a jeho následná přesnost 5.1.

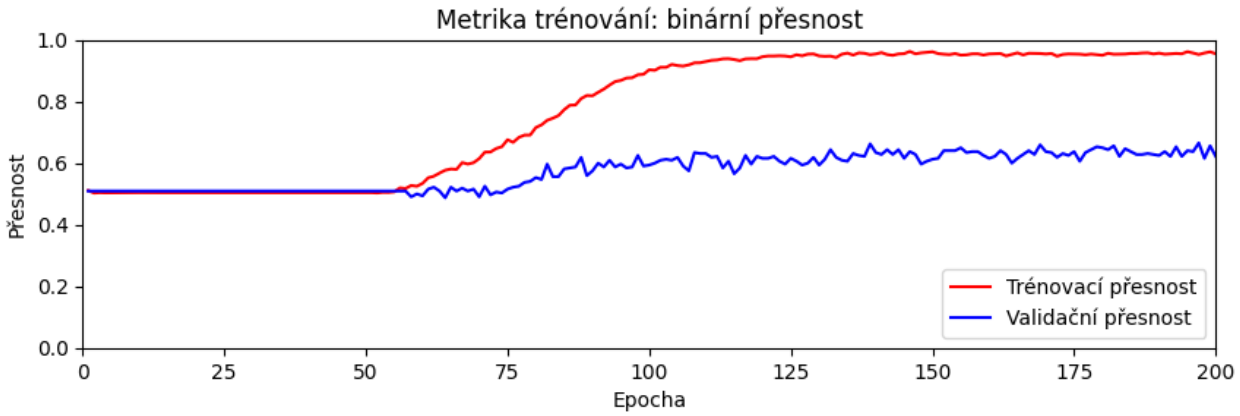


Obrázek 5.1: Průběh trénování LSTM modelu v zobrazení metriky binární přesnosti

```
[Fáze 1] LSTM Přesnost: 0.609
[Fáze 2] LSTM Přesnost: 0.633
```

Listing 5.1: Výsledek LSTM modelu na anglickém datasetu po trénování 5.1

Graf trénování 5.2 LSTM s technikou GloVe modelu a jeho následná přesnost 5.2.

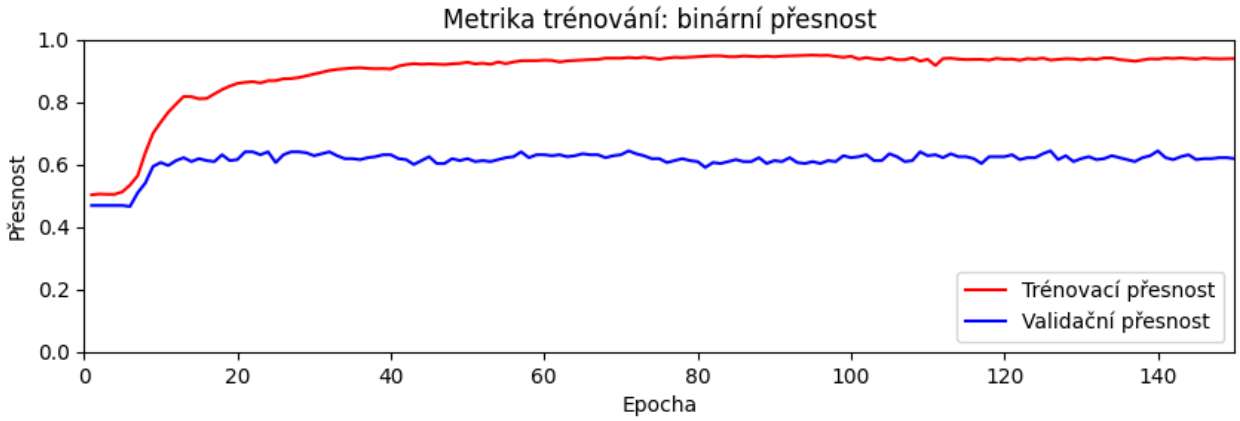


Obrázek 5.2: Průběh trénování LSTM GloVe modelu v zobrazení metriky binární přesnosti

```
[Fáze 1] LSTM GloVe Přesnost: 0.582
[Fáze 2] LSTM GloVe Přesnost: 0.615
```

Listing 5.2: Výsledek GloVe LSTM modelu na anglickém datasetu po trénování 5.2

Graf trénování 5.3 LSTM s technikou Attention modelu a jeho následná přesnost 5.3.

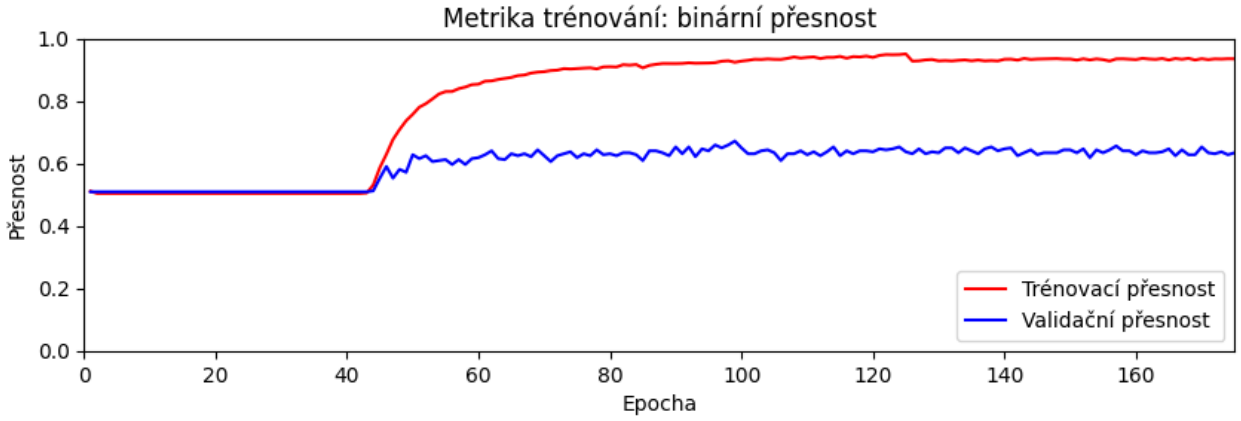


Obrázek 5.3: Průběh trénování LSTM Attention modelu v zobrazení metriky binární přesnosti

```
[Fáze 1] LSTM Attention Přesnost: 0.631
[Fáze 2] LSTM Attention Přesnost: 0.633
```

Listing 5.3: Výsledek LSTM Attention modelu na anglickém datasetu po trénování 5.3

Graf trénování 5.4 GRU modelu a jeho následná přesnost 5.4.

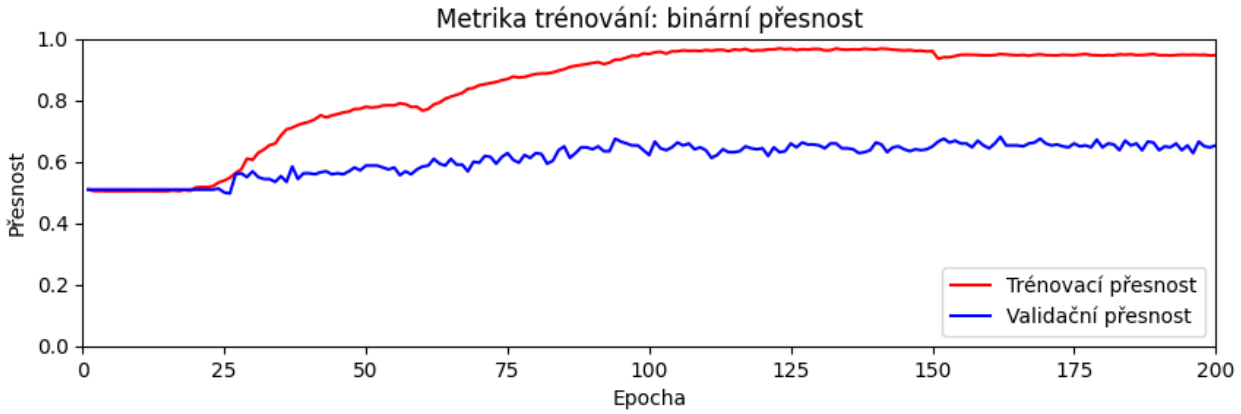


Obrázek 5.4: Průběh trénování GRU modelu v zobrazení metriky binární přesnosti

```
[Fáze 1] GRU Přesnost: 0.598
[Fáze 2] GRU Přesnost: 0.606
```

Listing 5.4: Výsledek GRU modelu na anglickém datasetu po trénování 5.4

Graf trénování 5.5 GRU s technikou GloVe modelu a jeho následná přesnost 5.5.

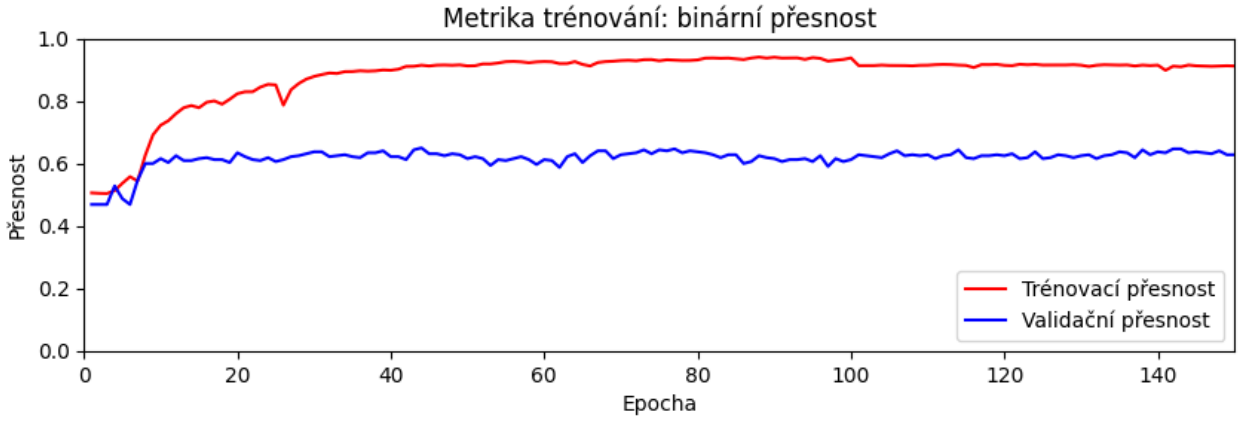


Obrázek 5.5: Průběh trénování GRU modelu s technikou GloVe v zobrazení metriky binární přesnosti

```
[Fáze 1] GRU GloVe Přesnost: 0.617
[Fáze 2] GRU GloVe Přesnost: 0.647
```

Listing 5.5: Výsledek GRU modelu s technikou GloVe na anglickém datasetu po trénování 5.5

Graf trénování 5.6 GRU s technikou Attention modelu a jeho následná přesnost 5.6.

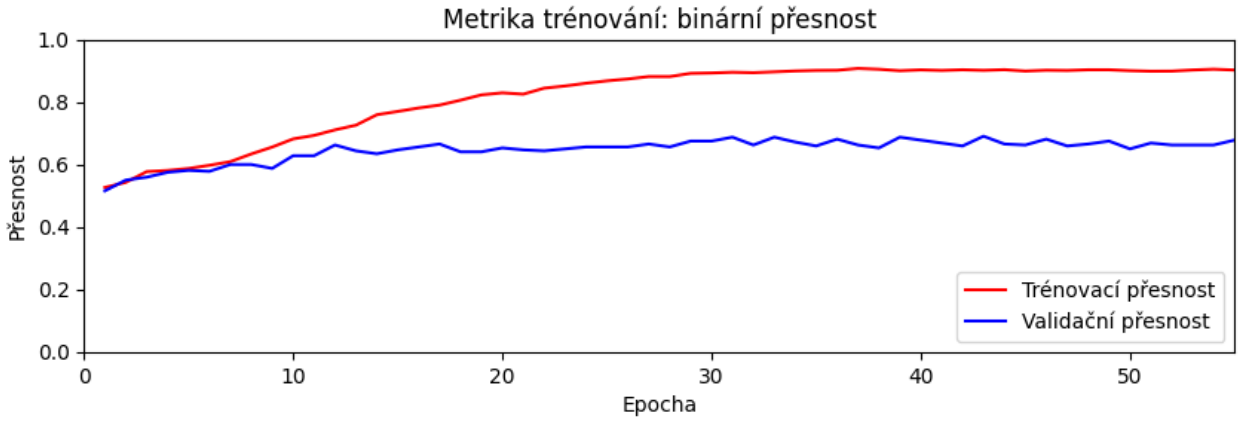


Obrázek 5.6: Průběh trénování GRU modelu s technikou Attention v zobrazení metriky binární přesnosti

```
[Fáze 1] GRU Attention Přesnost: 0.590
[Fáze 2] GRU Attention Přesnost: 0.639
```

Listing 5.6: Výsledek GRU modelu s technikou Attention na anglickém datasetu po trénování 5.6

Graf trénování 5.7 BERT modelu a jeho následná přesnost 5.7.



Obrázek 5.7: Průběh trénování BERT modelu v zobrazení metriky binární přesnosti

```
[Fáze 1] BERT Přesnost: 0.652
[Fáze 2] BERT Přesnost: 0.660
```

Listing 5.7: Výsledek BERT modelu na anglickém datasetu po trénování 5.7

Podle celkové tabulky 5.10 dosáhl nejlepší přesnosti klasifikátor BERT. Klasifikátory BERT, LSTM s technikou Attention a GRU s technikou Attention projevovaly schopnost učení již od brzkého začátku trénování na trénovacích datech. Naopak klasifikátory LSTM, GRU, LSTM s technikou

GloVe a GRU s technikou GloVe vykazovaly pomalé počáteční učení. Každopádně u všech klasifikátorů se postupem času zlepšovala jak trénovací, tak validační přesnost, což naznačuje postupné zdokonalování modelu při učení.

Tabulka 5.10: Trénování klasifikátorů

Klasifikátor	dosažená přesnost v první fázi	dosažená přesnost ve druhé fázi
LSTM	61 %	63 %
LSTM s technikou GloVe	58 %	62 %
LSTM s technikou Attention	63 %	63 %
GRU	60 %	61 %
GRU s technikou GloVe	62 %	65 %
GRU s technikou Attention	59 %	64 %
BERT	**65 %**	**66 %**

5.2.8.2 Počet trénovacích epoch

Klasifikátory s použitím stejné techniky mají tendenci se učit podobným způsobem, což je patrné z tabulky epoch klasifikátorů 5.11. Klasifikátor BERT prošel nejméně učícími epochami ze všech ostatních klasifikátorů a dosáhl nejlepšího učení viz. 5.10.

Tabulka 5.11: Trénování klasifikátorů

Klasifikátor	počet epoch první fáze	počet epoch druhé fázi
LSTM	125	50
LSTM s technikou GloVe	150	50
LSTM s technikou Attention	100	50
GRU	125	50
GRU s technikou GloVe	150	50
GRU s technikou Attention	100	50
BERT	**50**	**50**

5.2.8.3 Doba trénování

Průměrná doba trénování klasifikátorů 5.12 je přibližně 9 minut. Je však třeba poznamenat, že klasifikátor BERT představuje výjimku, neboť i přes nejmenší počet učících epoch se trénoval nejdéle ze všech klasifikátorů, konkrétně 17 minut a 4 vteřiny. Nejrychleji učícím se klasifikátorem je GRU s Attention technikou.

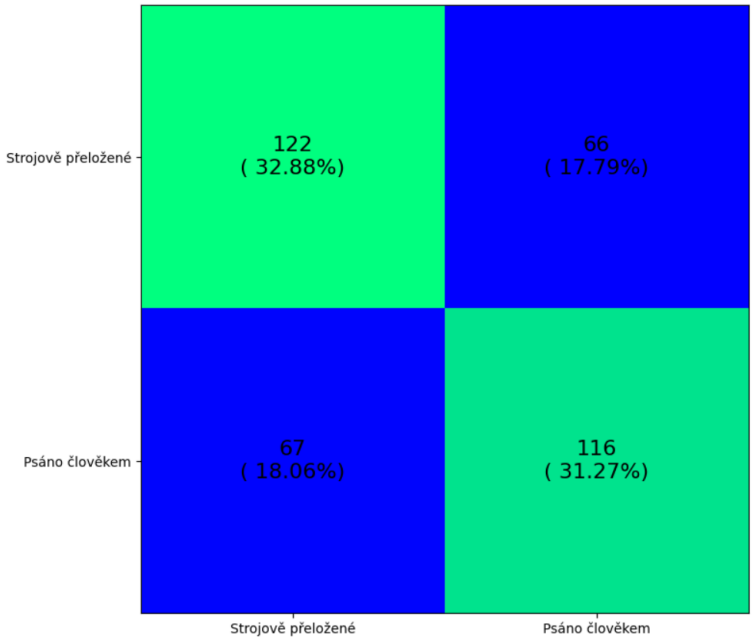
Tabulka 5.12: Doba trénování klasifikátorů

Klasifikátor	doba trénování první fáze	doba trénování druhé fázi	celková doba trénován
LSTM	373 s	167 s	541 s
LSTM s technikou GloVe	493 s	168 s	661 s
LSTM s technikou Attention	287 s	138 s	426 s
GRU	406 s	160 s	566 s
GRU s technikou GloVe	483 s	161 s	644 s
GRU s technikou Attention	**268 s**	**127 s**	**396 s**
BERT	565 s	459 s	1024 s

5.2.8.4 Prioritizace predikce

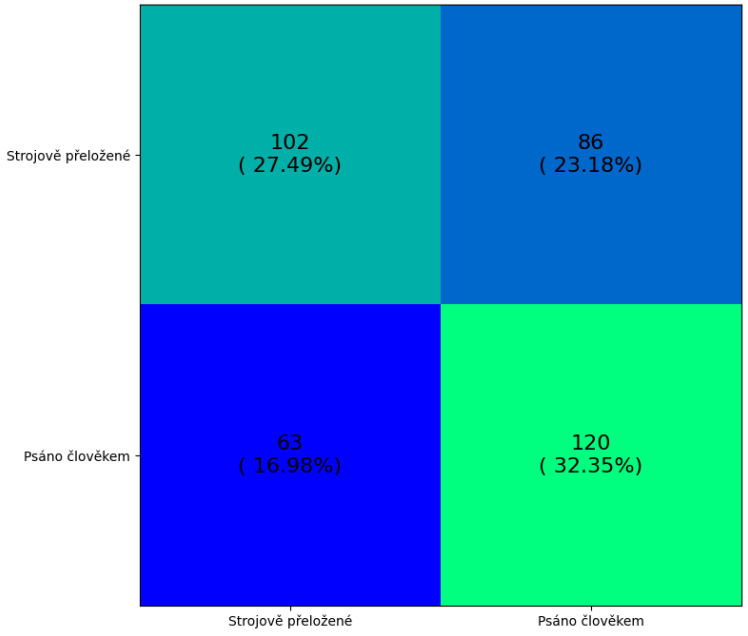
V následujících confusion maticích 5.8 5.9 5.10 5.11 5.12 5.13 5.14 jsou zobrazeny predikce modelu ve čtyřech kategoriích: správně pozitivní (true positive), chybně pozitivní (false positive), chybně negativní (false negative) a správně negativní (true negative). Ideálním scénářem je, když model dosahuje co nejvíce správně pozitivních a správně negativních předpovědí. Jakákoli preference jedné z možných odpovědí (psáno člověkem nebo strojově přeložené) představuje chybné naučení modelu. Cílem je dosáhnout vyváženého a správného předpovídání bez přehnaného zkreslení ve prospěch jedné kategorie. Pouze takový model je schopný poskytovat spolehlivé výsledky a má skutečnou hodnotu při klasifikaci.

Graf confusion matice 5.8 pro klasifikátor LSTM.



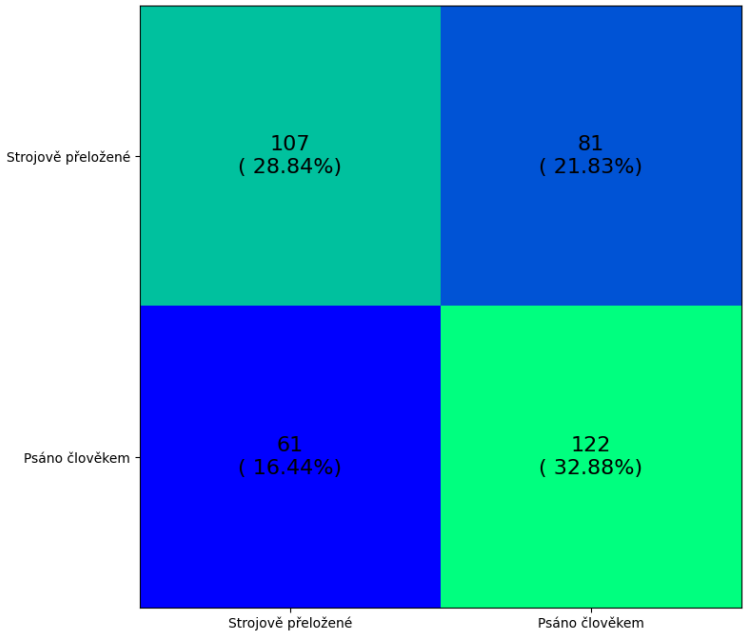
Obrázek 5.8: Výsledné predikování LSTM modelu na testovacích datech zobrazené v confusion matrix

Graf confusion matice 5.9 pro klasifikátor LSTM s technikou GloVe.



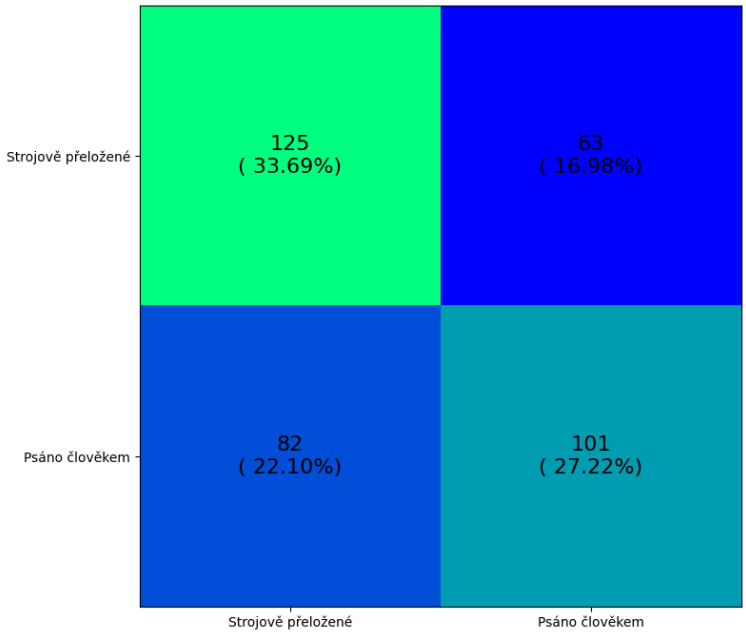
Obrázek 5.9: Výsledné predikování LSTM s technikou GloVe modelu na testovacích datech zobrazené v confusion matrix

Graf confusion matice 5.10 pro klasifikátor LSTM s technikou Attention.



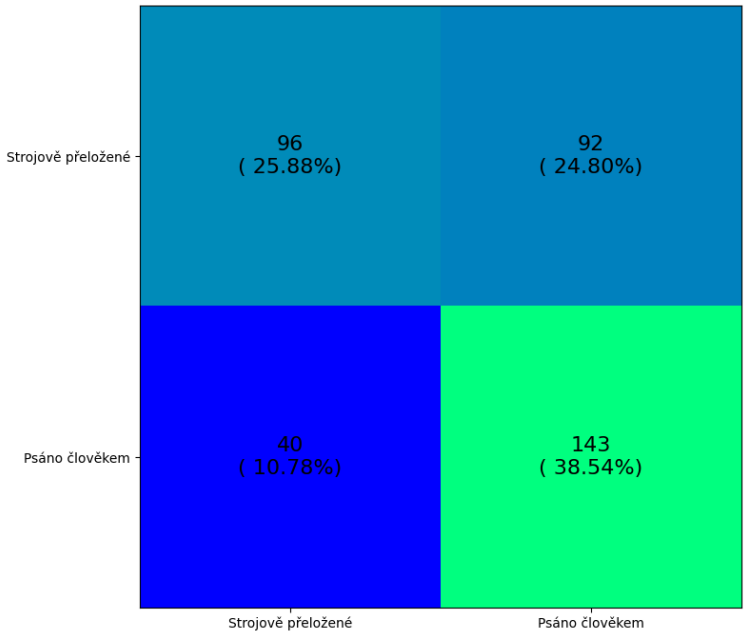
Obrázek 5.10: Výsledné predikování LSTM s technikou Attention modelu na testovacích datech zobrazené v confusion matrix

Graf confusion matice 5.11 pro klasifikátor GRU.



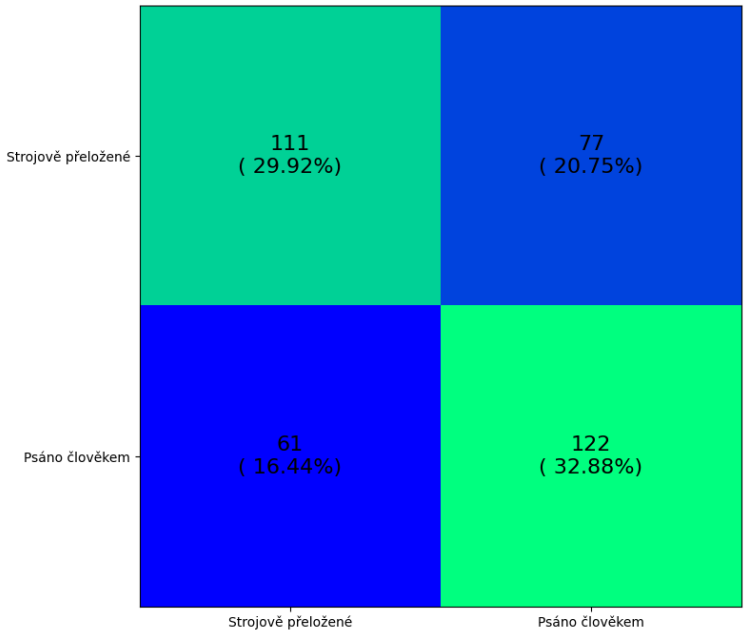
Obrázek 5.11: Výsledné predikování GRU modelu na testovacích datech zobrazené v confusion matrix

Graf confusion matice 5.12 pro klasifikátor GRU s technikou GloVe.



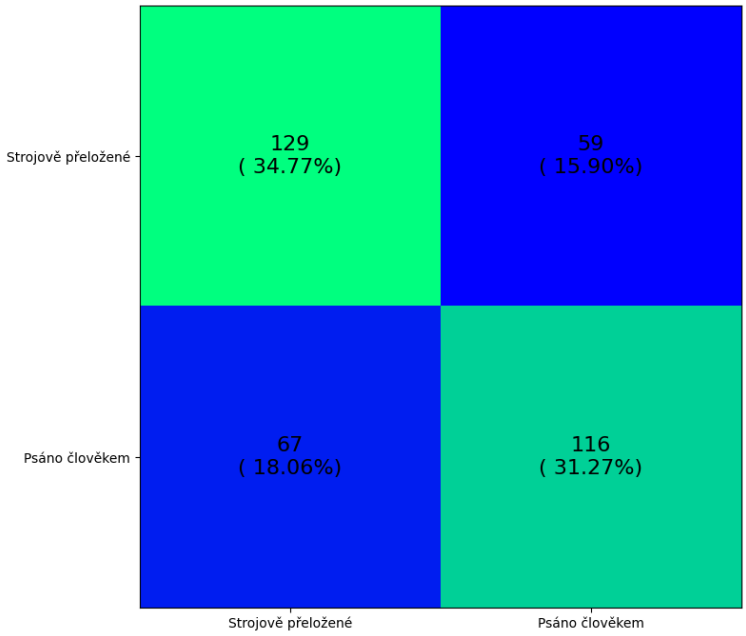
Obrázek 5.12: Výsledné predikování GRU s technikou GloVe modelu na testovacích datech zobrazené v confusion matrix

Graf confusion matice 5.13 pro klasifikátor GRU s technikou Attention.



Obrázek 5.13: Výsledné predikování GRU s technikou Attention modelu na testovacích datech zobrazené v confusion matrix

Graf confusion matice 5.14 pro klasifikátor BERT.



Obrázek 5.14: Výsledné predikování BERT modelu na testovacích datech zobrazené v confusion matrix

Většina klasifikátorů předpovídá vyváženě jak pro detekci strojově přeložených, tak i člověkem psaných vět. Nicméně klasifikátor GRU s technikou GloVe se od ostatních klasifikátorů liší a vykazuje větší tendenci k predikci vět psaných člověkem.

5.2.8.5 Predikce na reálných datech

Pro reálná data byla vybrána věta 'I am a very smart robot writting the message'. V následující tabulce 5.13 jsou uvedeny predikce jednotlivých klasifikátorů. Jelikož tato věta byla psána autorem práce, správnou predikcí je 'psáno člověkem'.

Tabulka 5.13: Predikce na reálné větě

Klasifikátor	predikční skóre	interpretace skóre
LSTM	**0.98337**	**Psáno člověkem**
LSTM s technikou GloVe	0.01706	Strojově přeloženo
LSTM s technikou Attention	**0.99378**	**Psáno člověkem**
GRU	0.07115	Strojově přeloženo
GRU s technikou GloVe	**0.99479**	**Psáno člověkem**
GRU s technikou Attention	**0.92976**	**Psáno člověkem**
BERT	0.00019	Strojově přeloženo

Pouze klasifikátory LSTM, LSTM s technikou attention, GRU s technikou GloVe a GRU s technikou Attention byly schopny správně predikovat odpověď.

5.3 Výsledné klasifikátory více jazyčného datasetu

Klasifikátory, určené pro anglická a vícejazyčná data, sdílí podobné architektury a využívají podobné techniky a strukturu sítě. Tento přístup, kdy jsou osvědčené architektury a techniky aplikovány na různá jazyková prostředí s minimálními úpravami, je často praktikován. Tím je umožněno využít předchozích znalostí a zkušeností s architekturou a přenést je na nová jazyková prostředí. Tímto způsobem je dosaženo efektivního využití existujících modelů a minimalizace potřeby vytvářet nové architektury pro každý jazyk zvlášť.

5.3.1 Architektura klasifikátoru LSTM

Tento klasifikátor 5.14 je identický svému anglickému protějšku 5.2.1, s výjimkou počtu parametrů. Celkový počet parametrů modelu činí 5 861 761.

Tabulka 5.14: Více jazyčný LSTM model

Vrstva modelu	výstupný formát dat	počet parametrů
TextVectorization	(None, 256)	0
Embedding	(None, 256, 128)	5 120 000
Bidirectional	(None, 256, 256)	263 168
Bidirectional	(None, 256)	394 240
Dense	(None, 256)	65 792
Gaussian Noise	(None, 256)	0
Dense	(None, 64)	16 448
Gaussian Noise	(None, 64)	0
Dense	(None, 32)	2 080
Dropout	(None, 32)	0
Dense	(None, 1)	33

5.3.2 Architektura klasifikátoru GRU

Tento klasifikátor 5.15 je identický svému anglickému protějšku 5.2.2, s výjimkou počtu parametrů. Celkový počet parametrů modelu činí 5 698 945.

Tabulka 5.15: Více jazyčný GRU model

Vrstva modelu	výstupný formát dat	počet parametrů
TextVectorization	(None, 256)	0
Embedding	(None, 256, 128)	5 120 000
Bidirectional	(None, 256, 256)	198 144
Bidirectional	(None, 256)	296 448
Dense	(None, 256)	65 792
Gaussian Noise	(None, 256)	0
Dense	(None, 64)	16 448
Gaussian Noise	(None, 64)	0
Dense	(None, 32)	2 080
Dropout	(None, 32)	0
Dense	(None, 1)	33

5.3.3 Architektura klasifikátoru LSTM s technikou GloVe

Tento klasifikátor 5.16 je identický svému anglickému protějšku 5.2.3, s výjimkou počtu parametrů. Celkový počet parametrů modelu činí 8 815 889.

Tabulka 5.16: Více jazyčný LSTM model s technikou GloVe

Vrstva modelu	výstupný formát dat	počet parametrů
TextVectorization	(None, 256)	0
GloVe	(None, 256, 200)	8 000 400
Bidirectional	(None, 256, 256)	336 896
Bidirectional	(None, 256)	394 240
Dense	(None, 256)	65 792
Gaussian Noise	(None, 256)	0
Dense	(None, 64)	16 448
Gaussian Noise	(None, 64)	0
Dense	(None, 32)	2 080
Gaussian Noise	(None, 32)	0
Dropout	(None, 32)	0
Dense	(None, 1)	33

5.3.4 Architektura klasifikátoru GRU s technikou GloVe

Tento klasifikátor 5.17 je identický svému anglickému protějšku 5.2.4, s výjimkou počtu parametrů. Celkový počet parametrů modelu činí 8 634 641.

Tabulka 5.17: Více jazyčný GRU model s technikou GloVe

Vrstva modelu	výstupný formát dat	počet parametrů
TextVectorization	(None, 256)	0
GloVe	(None, 256, 200)	8 000 400
Bidirectional	(None, 256, 256)	336 896
Bidirectional	(None, 256)	394 240
Dense	(None, 256)	65 792
Gaussian Noise	(None, 256)	0
Dense	(None, 64)	16 448
Gaussian Noise	(None, 64)	0
Dense	(None, 32)	2 080
Gaussian Noise	(None, 32)	0
Dropout	(None, 32)	0
Dense	(None, 1)	33

5.3.5 Architektura klasifikátoru LSTM s technikou Attention

Tento klasifikátor 5.18 je identický svému anglickému protějšku 5.2.5, s výjimkou počtu parametrů. Celkový počet parametrů modelu nyní činí 5 862 273.

Tabulka 5.18: Více jazyčný LSTM model s technikou Attention

Vrstva modelu	výstupný formát dat	počet parametrů
Text Vectorization	(None, 256)	0
Embedding	(None, 256, 128)	5 120 000
Bidirectional	(None, 256, 256)	263 168
Gaussian Noise	(None, 256, 256)	0
Bidirectional	(None, 256, 256)	394 240
Gaussian Noise	(None, 256, 256)	0
Attention	(None, 256)	512
Dense	(None, 256)	65 792
Gaussian Noise	(None, 256)	0
Dense	(None, 64)	16 448
Gaussian Noise	(None, 64)	0
Dense	(None, 32)	2 080
Gaussian Noise	(None, 32)	0
Dropout	(None, 32)	0
Dense	(None, 1)	33

5.3.6 Architektura klasifikátoru GRU s technikou Attention

Tento klasifikátor 5.19 je identický svému anglickému protějšku 5.2.6, s výjimkou počtu parametrů. Celkový počet parametrů modelu nyní činí 5 699 457.

Tabulka 5.19: Více jazyčný GRU model s technikou Attention

Vrstva modelu	výstupný formát dat	počet parametrů
Text Vectorization	(None, 256)	0
Embedding	(None, 256, 128)	5 120 000
Bidirectional	(None, 256, 256)	198 144
Gaussian Noise	(None, 256, 256)	0
Bidirectional	(None, 256, 256)	296 448
Gaussian Noise	(None, 256, 256)	0
Attention	(None, 256)	512
Dense	(None, 256)	65 792
Gaussian Noise	(None, 256)	0
Dense	(None, 64)	16 448
Gaussian Noise	(None, 64)	0
Dense	(None, 32)	2 080
Gaussian Noise	(None, 32)	0
Dropout	(None, 32)	0
Dense	(None, 1)	33

5.3.7 Architektura klasifikátoru BERT

Tento klasifikátor 5.20 je identický svému anglickému protějšku 5.2.7. Celkový počet parametrů modelu činí 28 764 162.

Tabulka 5.20: Více jazyčný BERT model

Vrstva modelu	výstupný formát dat	počet parametrů
InputLayer	(None,)	0
Preprocessing	—	0
BERT encoder	—	28,763,649
Dropout	(None, 512)	0
Dense	(None, 1)	513

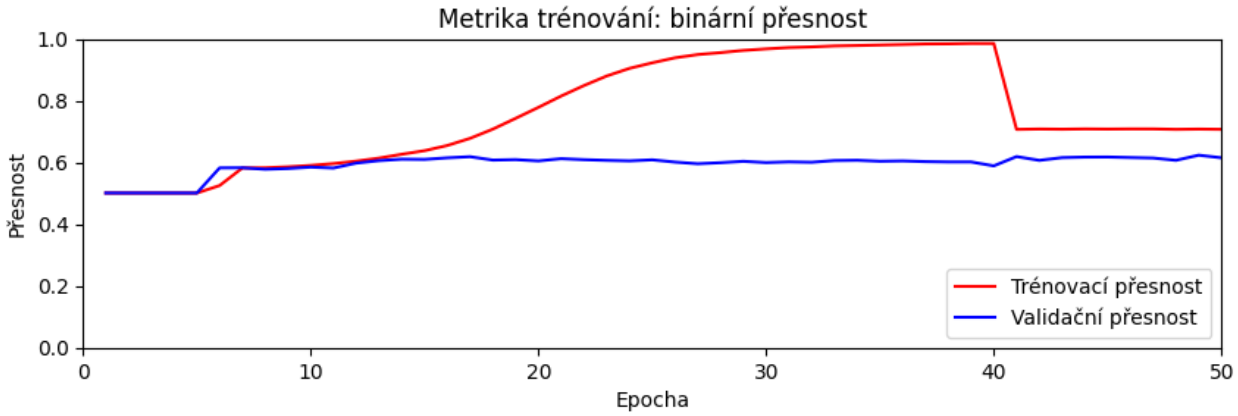
5.3.8 Výsledky klasifikátorů

5.3.8.1 Trénovací fáze

V následujících grafech 5.15 5.16 5.17 5.18 5.19 5.20 5.21 jsou znázorněny křivky učení v průběhu postupně vykonávaných epoch. Červená křivka představuje přesnost predikce modelu na trénovacích

datech a modrá křivka představuje přesnost predikce modelu na validačních datech.

Graf trénování 5.15 LSTM modelu a jeho následná přesnost 5.8.

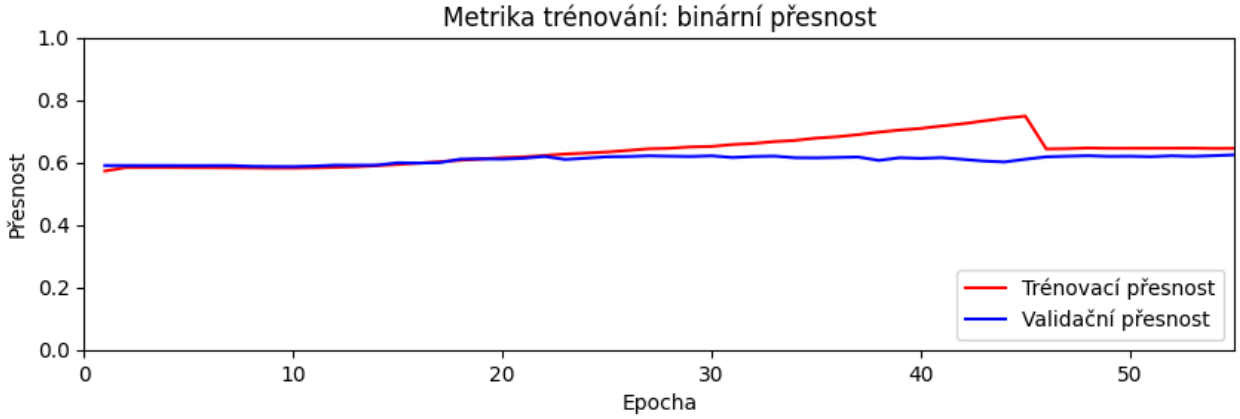


Obrázek 5.15: Průběh trénování LSTM modelu v zobrazení metriky binární přesnosti

```
[Fáze 1] LSTM Přesnost: 0.606
[Fáze 2] LSTM Přesnost: 0.606
```

Listing 5.8: Výsledek LSTM modelu na více jazyčném datasetu po trénování 5.15

Graf trénování 5.16 LSTM s technikou GloVe modelu a jeho následná přesnost 5.9.

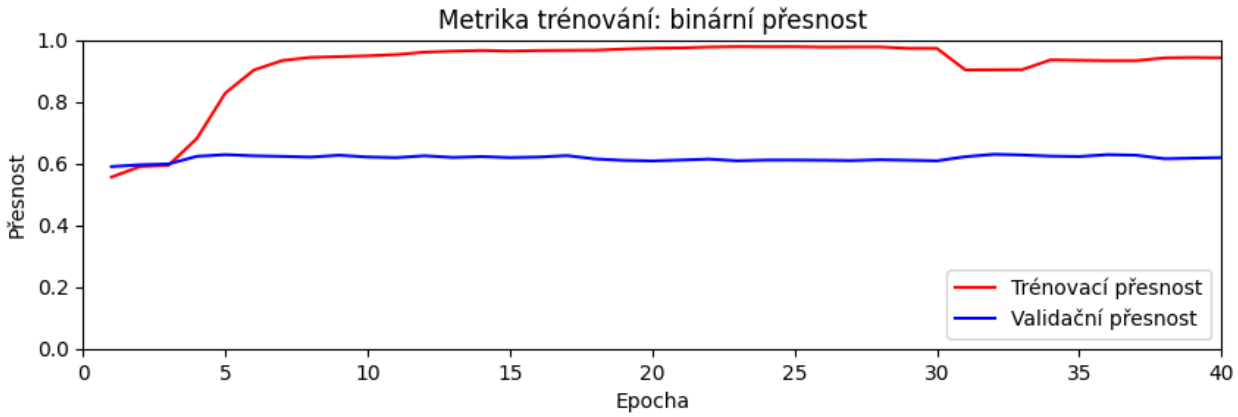


Obrázek 5.16: Průběh trénování LSTM GloVe modelu v zobrazení metriky binární přesnosti

```
[Fáze 1] LSTM GloVe Přesnost: 0.612
[Fáze 2] LSTM GloVe Přesnost: 0.612
```

Listing 5.9: Výsledek GloVe LSTM modelu na více jazyčném datasetu po trénování 5.16

Graf trénování 5.17 LSTM s technikou Attention modelu a jeho následná přesnost 5.10.

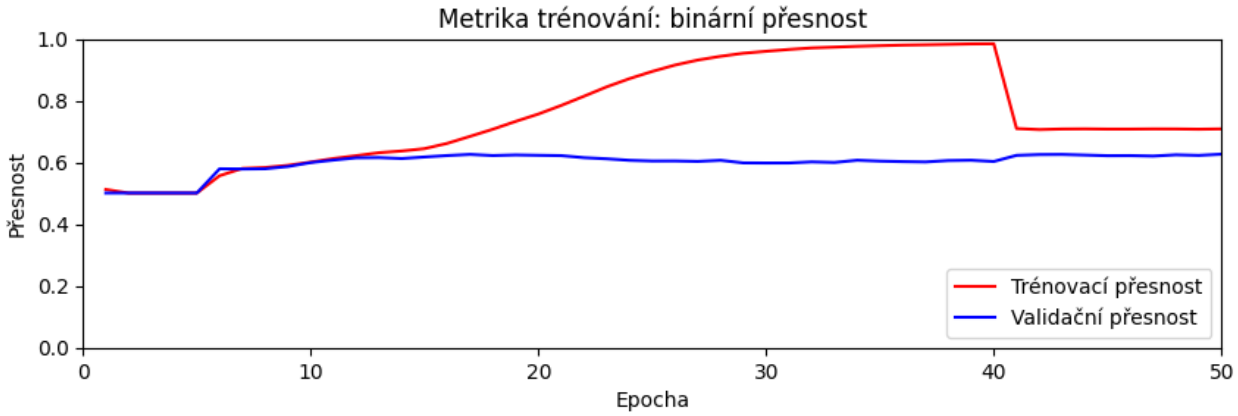


Obrázek 5.17: Průběh trénování LSTM Attention modelu v zobrazení metriky binární přesnosti

```
[Fáze 1] LSTM Attention Přesnost: 0.597
[Fáze 2] LSTM Attention Přesnost: 0.600
```

Listing 5.10: Výsledek LSTM Attention modelu na více jazyčném datasetu po trénování 5.17

Graf trénování 5.18 GRU modelu a jeho následná přesnost 5.11.

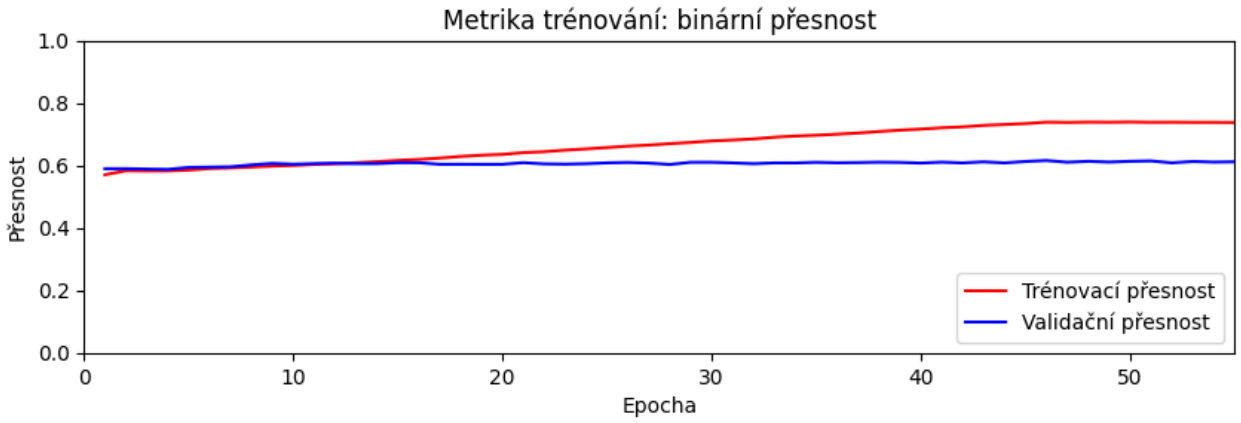


Obrázek 5.18: Průběh trénování GRU modelu v zobrazení metriky binární přesnosti

```
[Fáze 1] GRU Přesnost: 0.611
[Fáze 2] GRU Přesnost: 0.611
```

Listing 5.11: Výsledek GRU modelu na více jazyčném datasetu po trénování 5.18

Graf trénování 5.19 GRU s technikou GloVe modelu a jeho následná přesnost 5.12.

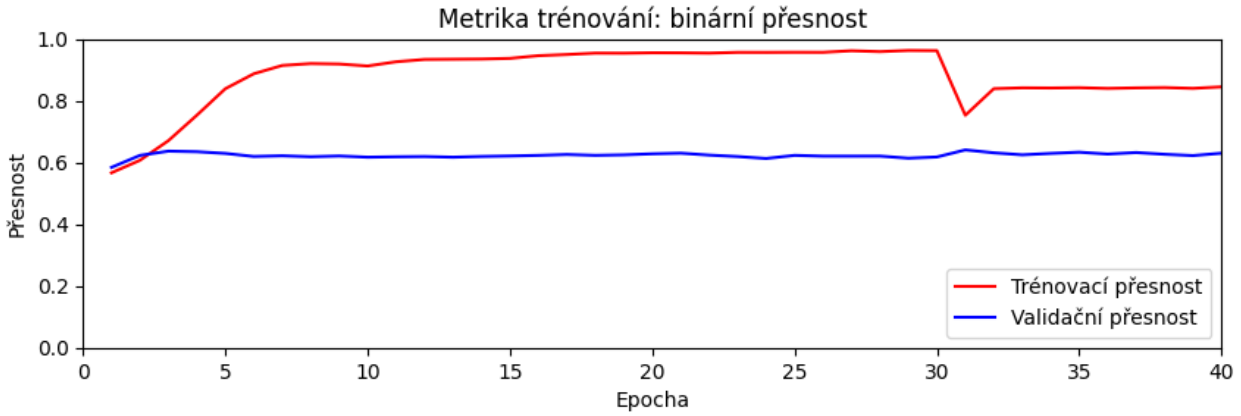


Obrázek 5.19: Průběh trénování GRU s technikou GloVe modelu v zobrazení metriky binární přesnosti

```
[Fáze 1] GRU GloVe Přesnost: 0.595
[Fáze 2] GRU GloVe Přesnost: 0.595
```

Listing 5.12: Výsledek GRU s technikou GloVe modelu na více jazyčném datasetu po trénování 5.19

Graf trénování 5.20 GRU s technikou Attention modelu a jeho následná přesnost 5.13.

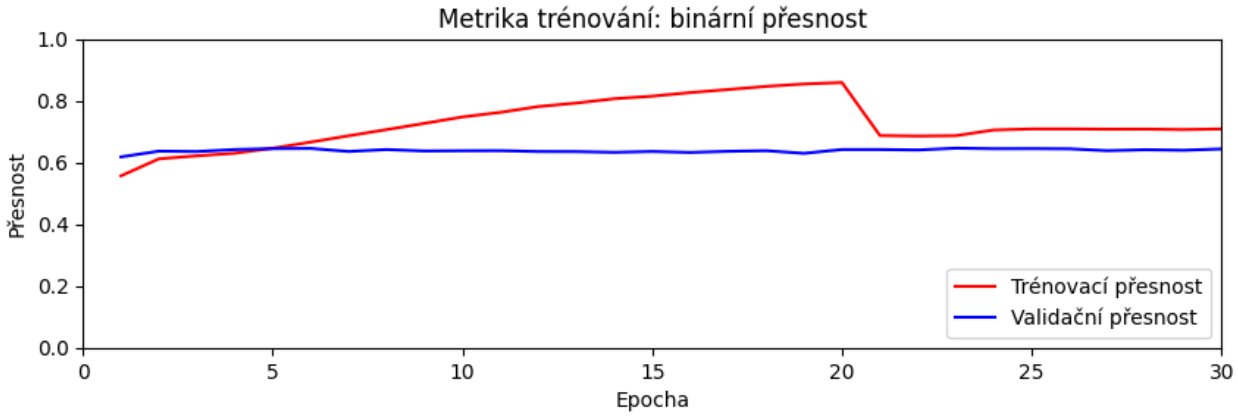


Obrázek 5.20: Průběh trénování GRU modelu s technikou Attention v zobrazení metriky binární přesnosti

```
[Fáze 1] GRU Attention Přesnost: 0.628
[Fáze 2] GRU Attention Přesnost: 0.630
```

Listing 5.13: Výsledek GRU modelu s technikou Attention na více jazyčném datasetu po trénování 5.20

Graf trénování 5.21 BERT modelu a jeho následná přesnost 5.14.



Obrázek 5.21: Průběh trénování BERT modelu v zobrazení metriky binární přesnosti

```
[Fáze 1] BERT Přesnost: 0.626
[Fáze 2] BERT Přesnost: 0.627
```

Listing 5.14: Výsledek BERT modelu na více jazyčném datasetu po trénování 5.21

Podle celkové tabulky 5.21 dosáhly nejlepší přesnosti klasifikátory BERT a klasifikátor GRU s technikou Attention. Klasifikátory LSTM a GRU vykazovaly pomalé počáteční učení, zatímco ostatní klasifikátory se rychleji naučily. U většiny klasifikátorů se druhá fáze učení neprojevila jako výrazné zlepšení, s výjimkou klasifikátorů BERT, LSTM s technikou Attention a GRU s technikou Attention, které dosáhly lepší konfigurace modelu.

Grafy ukazují výrazné skoky ve trénovací přesnosti, což je způsobeno zahájením druhé fáze učení, kdy je vybrána nejlepší konfigurace na základě přesnosti na validačních datech. Tyto skoky signalizují výběr konfigurace z počátku trénování, kdy byla hodnota trénovací přesnosti ještě nízká. V tomto případě lze pozorovat, jak druhá fáze učení eliminuje přeučení na trénovacích datech.

Tabulka 5.21: Trénování klasifikátorů

Klasifikátor	dosažená přesnost v první fázi	dosažená přesnost ve druhé fázi
LSTM	61 %	61 %
LSTM s technikou GloVe	61 %	61 %
LSTM s technikou Attention	60 %	60 %
GRU	61 %	61 %
GRU s technikou GloVe	60 %	60 %
GRU s technikou Attention	**63** %	**63** %
BERT	**63 %**	**63 %**

5.3.8.2 Počet trénovacích epoch

Podle tabulky epoch klasifikátorů 5.22 je patrné, že klasifikátory využívající stejnou techniku vykazují tendenci učit se podobným způsobem. Klasifikátor BERT prošel nejméně epochami z celé sady a dosáhl vyšší přesnosti než průměr mezi ostatními klasifikátory.

Tabulka 5.22: Trénování klasifikátorů

Klasifikátor	počet epoch první fáze	počet epoch druhé fázi
LSTM	40	10
LSTM s technikou GloVe	45	10
LSTM s technikou Attention	30	10
GRU	40	10
GRU s technikou GloVe	45	10
GRU s technikou Attention	30	10
BERT	**20**	**10**

5.3.8.3 Doba trénování

Průměrná doba trénování klasifikátorů 5.23 je přibližně 14 minut. Je však třeba poznamenat, že klasifikátor BERT představuje výjimku, neboť i přes nejmenší počet učících epoch se trénoval nejdéle ze všech klasifikátorů, konkrétně 2 hodiny 30 minut a 49 vteřin. Nejrychleji učící se klasifikátory jsou LSTM s Attention technikou a GRU s Attention technikou.

Tabulka 5.23: Doba trénování klasifikátorů

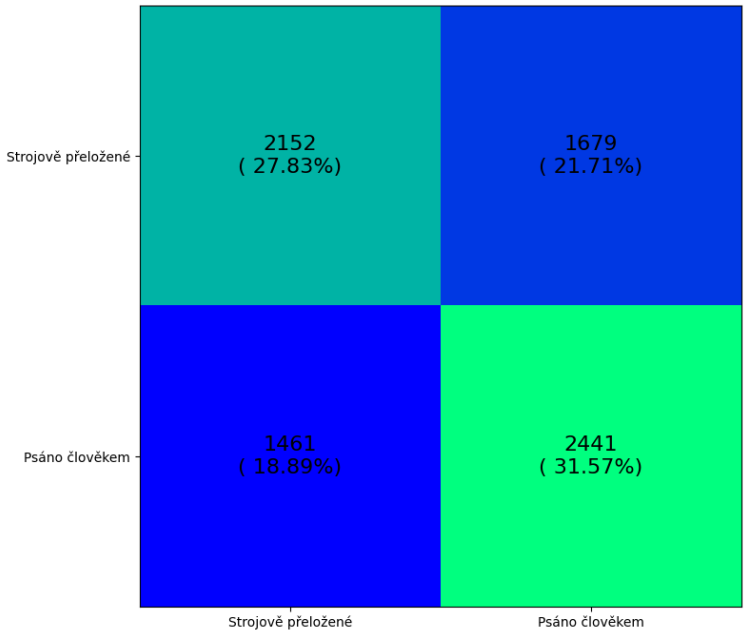
Klasifikátor	doba trénování první fáze	doba trénování druhé fázi	celková doba trénová
LSTM	685 s	166 s	851 s
LSTM s technikou GloVe	745 s	161 s	907 s
LSTM s technikou Attention	**592 s**	**189 s**	**781 s**
GRU	660 s	155 s	816 s
GRU s technikou GloVe	774 s	163 s	937 s
GRU s technikou Attention	**584 s**	**196 s**	**780 s**
BERT	6032 s	3017 s	9049 s

5.3.8.4 Prioritizace predikce

V následujících confusion maticích 5.22 5.23 5.24 5.25 5.26 5.27 5.28 jsou zobrazeny predikce modelu ve čtyřech kategoriích: správně pozitivní (true positive), chybně pozitivní (false positive), chybně

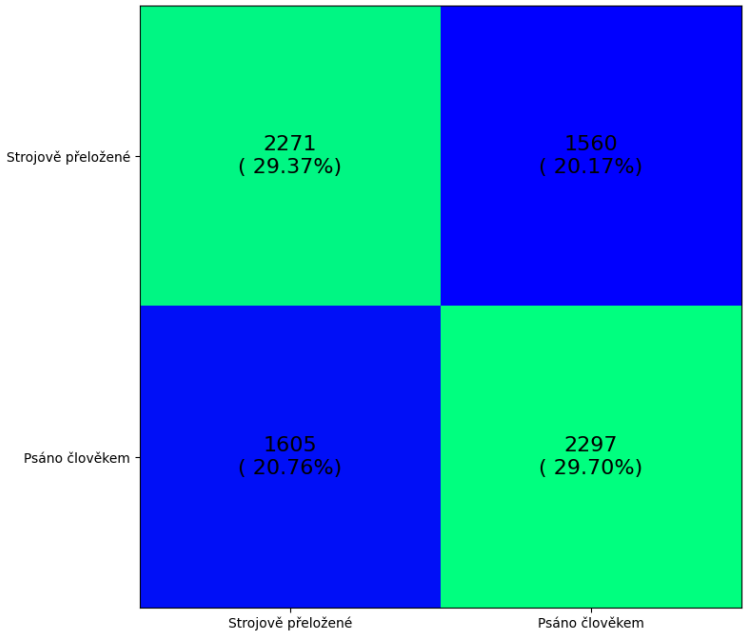
negativní (false negative) a správně negativní (true negative). Ideálním scénářem je, když model dosahuje co nejvíce správně pozitivních a správně negativních předpovědí. Jakákoli preference jedné z možných odpovědí (psáno člověkem nebo strojově přeložené) představuje chybné naučení modelu. Cílem je dosáhnout vyváženého a správného předpovídání bez přehnaného zkreslení ve prospěch jedné kategorie. Pouze takový model je schopný poskytovat spolehlivé výsledky a má skutečnou hodnotu při klasifikaci.

Graf confusion matice 5.22 pro klasifikátor LSTM.



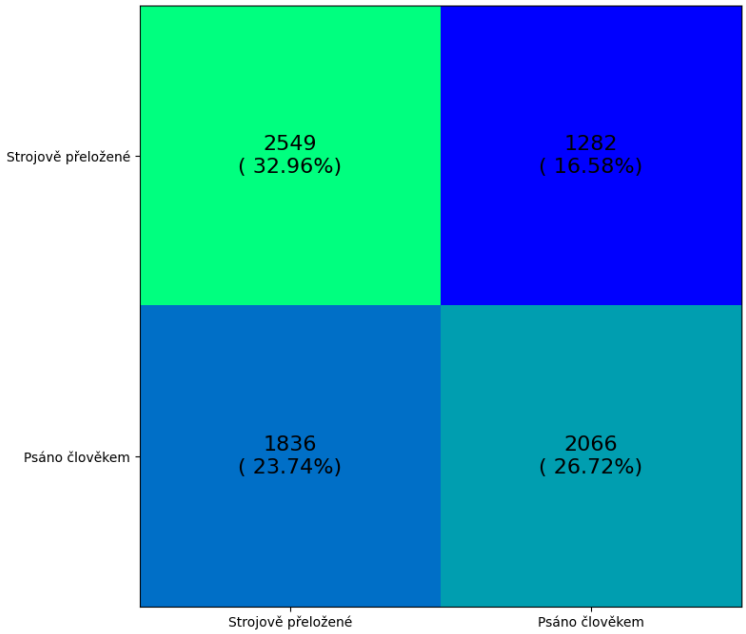
Obrázek 5.22: Výsledné predikování LSTM modelu na testovacích datech zobrazené v confusion matrix

Graf confusion matice 5.23 pro klasifikátor LSTM s technikou GloVe.



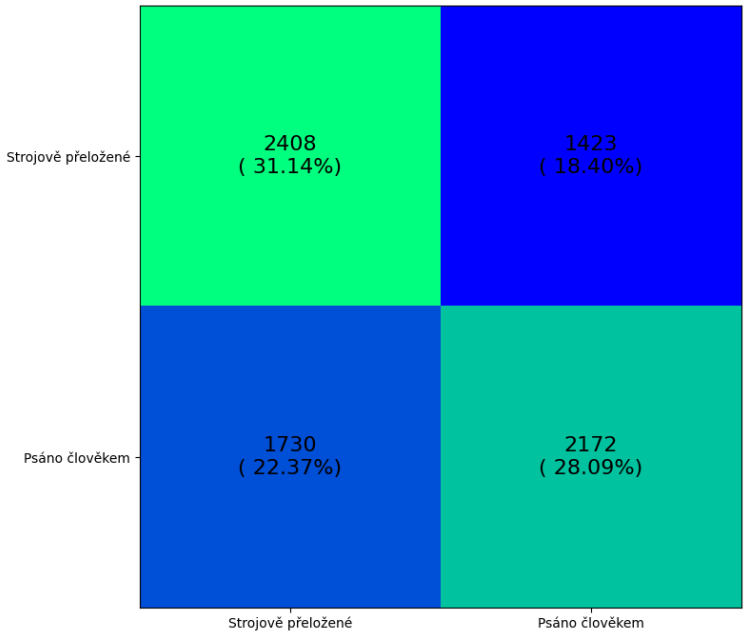
Obrázek 5.23: Výsledné predikování LSTM s technikou GloVe modelu na testovacích datech zobrazené v confusion matrix

Graf confusion matice 5.24 pro klasifikátor LSTM s technikou Attention.



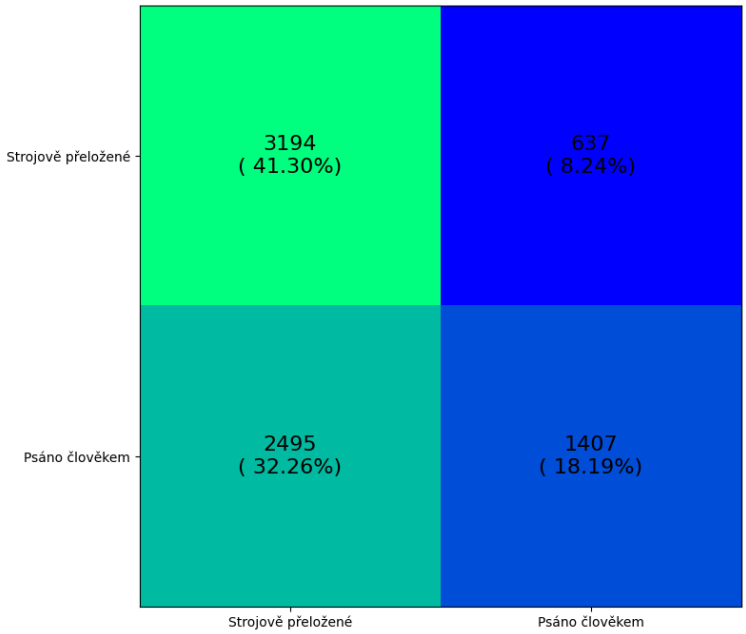
Obrázek 5.24: Výsledné predikování LSTM s technikou Attention modelu na testovacích datech zobrazené v confusion matrix

Graf confusion matice 5.25 pro klasifikátor GRU.



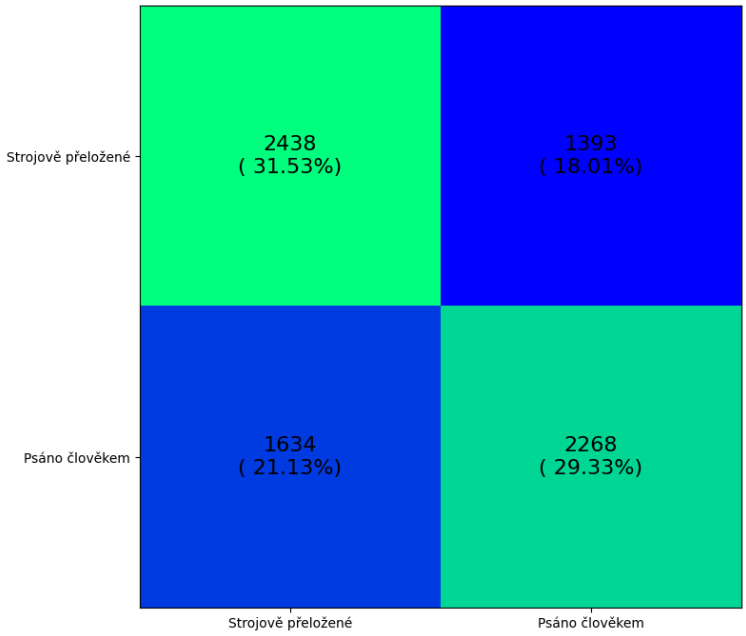
Obrázek 5.25: Výsledné predikování GRU modelu na testovacích datech zobrazené v confusion matrix

Graf confusion matice 5.26 pro klasifikátor GRU s technikou GloVe.



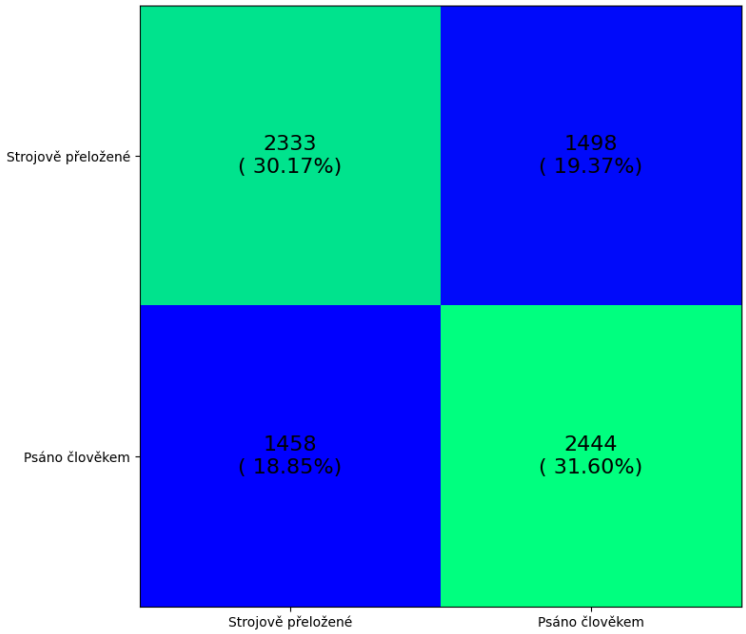
Obrázek 5.26: Výsledné predikování GRU s technikou GloVe modelu na testovacích datech zobrazené v confusion matrix

Graf confusion matice 5.27 pro klasifikátor GRU s technikou Attention.



Obrázek 5.27: Výsledné predikování GRU s technikou Attention modelu na testovacích datech zobrazené v confusion matrix

Graf confusion matice 5.28 pro klasifikátor BERT.



Obrázek 5.28: Výsledné predikování BERT modelu na testovacích datech zobrazené v confusion matrix

Většina klasifikátorů předpovídá vyváženě jak pro detekci strojově přeložených, tak i člověkem

psaných vět. Nicméně klasifikátor GRU s technikou GloVe se od ostatních klasifikátorů liší a vykazuje větší tendenci k predikci vět psaných člověkem.

5.3.8.5 Predikce na reálných datech

Pro reálná data byla vybrána věta 'I am a very smart robot writting the message'. V následující tabulce 5.24 jsou uvedeny predikce jednotlivých klasifikátorů. Jelikož tato věta byla psána autorem práce, správnou predikcí je 'psáno člověkem'.

Tabulka 5.24: Predikce na reálné větě

Klasifikátor	predikční skóre	interpretace skóre
LSTM	**0.59068**	**Psáno člověkem**
LSTM s technikou GloVe	**0.96997**	**Psáno člověkem**
LSTM s technikou Attention	**0.97166**	**Psáno člověkem**
GRU	**0.54896**	**Psáno člověkem**
GRU s technikou GloVe	0.42217	Strojově přeloženo
GRU s technikou Attention	**0.81786**	**Psáno člověkem**
BERT	**0.85771**	**Psáno člověkem**

Všechny klasifikátory s výjimkou GRU s technikou GloVe byly schopné správně předpovědět odpověď.

5.4 Porovnání klasifikátorů

Pro porovnání klasifikátorů budou použity tři kritéria, na jejichž základě se budou hodnotit klasifikátory. Tato kritéria jsou:

- Počet parametrů: Bude se měřit, kolik parametrů má každý klasifikátor. Tento ukazatel nám poskytne představu o složitosti a velikosti jednotlivých modelů.
- Doba trénování: Bude se sledovat, jak dlouho trvá trénování každého klasifikátoru. Toto kritérium nám umožní porovnat rychlost trénování a zjistit, který model se trénuje nejrychleji.
- Metrika přesnosti na testovacích datech: Bude se vyhodnocovat přesnost každého klasifikátoru na testovacích datech. Tím získáme informaci o schopnosti jednotlivých modelů správně klasifikovat neznámá data.

Na základě těchto tří kritérií budeme schopni porovnat a vyhodnotit jednotlivé klasifikátory.

5.4.1 Porovnání počtu parametrů

Obecně platí, že čím větší počet parametrů má model, tím větší je jeho kapacita a tím lépe by měl být schopen se naučit složitější úlohy. Parametry modelu představují váhy a biasy, které se optimalizují během trénování, aby model co nejlépe popsal vztahy mezi vstupy a výstupy.

Model s vyšším počtem parametrů má větší flexibilitu při aproximaci složitých funkcí a může se přizpůsobit složitějším vzorům ve vstupních datech. Avšak vyšší počet parametrů může také způsobit přetrénování, kdy se model naučí příliš specifickým detailům trénovacích dat a nemusí se dobře generalizovat na nová neznámá data.

Optimální počet parametrů závisí na konkrétní úloze a dostupných datových sadách. V některých případech mohou menší modely s nižším počtem parametrů dosahovat podobných výsledků jako velké modely, pokud je trénování provedeno správně a jsou použity vhodné metody regularizace.

Je důležité najít vyváženou rovnováhu mezi počtem parametrů a schopností modelu se naučit a generalizovat úlohu.

5.4.1.1 Porovnání počtu parametrů na anglickém datasetu

Průměrný počet parametrů klasifikátorů 5.25 se pohybuje kolem 2.7 milionů parametrů. Existuje však výjimka v podobě klasifikátoru BERT, který má přibližně 28 milionů parametrů. Tento rozdíl je způsoben složitostí a velikostí samotného modelu BERT.

Tabulka 5.25: Počty parametrů klasifikátorů

Klasifikátor	počet parametrů
LSTM	3 179 905
GRU	2 984 321
LSTM s technikou GloVe	2 604 009
GRU s technikou GloVe	2 422 761
LSTM s technikou Attention	3 180 417
GRU s technikou Attention	2 984 833
BERT	**28 764 162**

5.4.1.2 Porovnání počtu parametrů na více jazyčném datasetu

Většina klasifikátorů 5.26 má přibližně 5,7 milionů parametrů. Klasifikátory používající techniku GloVe mají přibližně 8,7 milionů parametrů. Klasifikátor BERT, s přibližně 28 miliony parametrů, má největší počet parametrů mezi všemi zkoumanými klasifikátory.

Tabulka 5.26: Počty parametrů klasifikátorů

Klasifikátor	počet parametrů
LSTM	5 861 761
GRU	5 698 945
LSTM s technikou GloVe	8 815 889
GRU s technikou GloVe	8 634 641
LSTM s technikou Attention	5 862 273
GRU s technikou Attention	5 699 457
BERT	**28 764 162**

5.4.2 Porovnání doby trénování

Doba trénování modelu závisí na jeho komplexnosti, dostupných výpočetních prostředcích a zvolené konfiguraci trénování. Je důležité správně nastavit parametry trénování, jako je velikost dávky (batch size), rychlost učení (learning rate), počet epoch a další hyperparametry, aby se dosáhlo optimálního výkonu modelu.

Není pravidlem, že co nejdéle trvající trénování vede k nejlepším výsledkům. Je zapotřebí správně naladit model a jeho hyperparametry a vyhnout se přetrénování (overfittingu) nebo podtrénování (underfittingu). To znamená, že model by měl být dostatečně komplexní, aby se naučil relevantní vzory ve vstupních datech, ale zároveň by měl být schopný generalizovat na nová data.

5.4.2.1 Klasifikátory trénovány na anglickém datasetu

Doba trénování různých klasifikátorů 5.27 tvoří podobné dvojice, kde techniky mají blízkou dobu trénování. Například v případě GloVe je rozdíl v době trénování mezi architekturami LSTM a GRU pouhých 17 vteřin.

Podobnost mezi těmito dvojicemi je způsobena nastavením počtu epoch pro trénování klasifikátorů. Když jsou stejné techniky natrénovány s podobným nebo identickým počtem epoch na trénovacích datech, dochází ke vzniku těchto podobností v době trénování.

Učení BERTa trvá nejdelší dobu z důvodu jeho složitosti a rozsáhlých parametrů 5.25. BERT se stal jedním z nejvýkonnějších modelů pro zpracování přirozeného jazyka, ale jeho trénování vyžaduje více času kvůli jeho komplexní struktuře a většímu počtu parametrů.

Tabulka 5.27: Celková doba trénování klasifikátorů

Klasifikátor	doba trénování
LSTM	541s
GRU	566s
LSTM s technikou GloVe	661s
GRU s technikou GloVe	644s
LSTM s technikou Attention	426s
GRU s technikou Attention	396s
BERT	**1024s**

5.4.2.2 Klasifikátory trénovány na více jazyčném datasetu

Doba trénování různých klasifikátorů 5.28 tvoří podobné dvojice, ze stejných důvodů jako pro anglický dataset viz. 5.4.2.1.

Rozdíl v době trénování mezi klasifikátory anglického datasetu a klasifikátory tohoto datasetu je způsoben větším rozsahem více jazyčného datasetu, na kterém museli být klasifikátory natrénovány. Větší datasety vyžadují delší dobu pro kompletní naučení modelu, jelikož je nutné zpracovat a analyzovat větší množství datových instancí. To může zpomalit celkový proces trénování a prodloužit dobu potřebnou k dosažení požadované úrovně výkonu klasifikátoru. Je důležité brát v úvahu velikost datasetu a jeho vliv na dobu trénování při srovnávání klasifikátorů.

Učení BERTa trvá nejdelší dobu z důvodu jeho složitosti a rozsáhlých parametrů 5.26.

Tabulka 5.28: Celková doba trénování klasifikátorů

Klasifikátor	doba trénování
LSTM	851s
GRU	816s
LSTM s technikou GloVe	907s
GRU s technikou GloVe	937s
LSTM s technikou Attention	781s
GRU s technikou Attention	780s
BERT	**9049s**

5.4.3 Porovnání v rámci metriky přesnosti nad testovacími daty

Metrika přesnosti je jednou z nejčastěji používaných metrik pro vyhodnocování přesnosti klasifikátorů. Metrika přesnosti udává, jaký podíl správně klasifikovaných instancí má model ze všech

klasifikovaných instancí.

V případě porovnání klasifikátorů pro detekci strojově přeložených a ručně psaných sekvencí textu, metrika přesnosti nám poskytne informaci o tom, jak dobře tyto klasifikátory rozlišují mezi těmito dvěma typy sekvencí. Vyšší přesnost znamená, že model správně identifikuje většinu strojově přeložených a ručně psaných sekvencí.

5.4.3.1 Klasifikátory trénovány na anglickém datasetu

Finální a nejdůležitější porovnání mezi klasifikátory je na základě metriky přesnosti 5.29. Všechny modely dosáhly přesnosti vyšší než 60 %, přičemž nejvyšší přesnost zaznamenal klasifikátor BERT s hodnotou 66 %.

Tento výsledek naznačuje, že BERT je nejefektivnějším klasifikátorem v daném kontextu. Je však také důležité vzít v úvahu další faktory, jako jsou rychlost trénování, náročnost implementace a případné omezení modelu v praxi. Nicméně z hlediska dosažené přesnosti se klasifikátor BERT ukazuje jako nejlepší volba mezi zkoumanými modely.

Tabulka 5.29: Metrika přesnosti klasifikátorů

Klasifikátor	přesnost
LSTM	63 %
GRU	60 %
LSTM s technikou GloVe	61 %
GRU s technikou GloVe	65 %
LSTM s technikou Attention	63 %
GRU s technikou Attention	64 %
BERT	**66 %**

5.4.3.2 Klasifikátory trénovány na více jazyčném datasetu

Finální a nejdůležitější porovnání mezi klasifikátory se provádí na základě metriky přesnosti 5.30. Všechny modely dosáhly přesnosti, která je buď stejná nebo vyšší než 60 %. Nejvyšší přesnost byla dosažena klasifikátorem BERT a klasifikátorem GRU s technikou attention, které oba dosáhly přesnosti 63 %.

Na základě informací o velikosti modelu 5.26, době trénování 5.28 a dosažené metrice přesnosti 5.30 lze usoudit, že nejlepším klasifikátorem je architektura GRU s použitím attention techniky, i přestože klasifikátor BERT dosáhl stejné přesnosti. Tento závěr je podpořen faktem, že klasifikátor BERT vyžadoval mnohem delší dobu trénování než GRU s technikou attention. Tedy GRU s attention technikou je schopný dosáhnout stejné přesnosti jako BERT v kratším čase, jelikož je méně komplexní, což jej činí efektivnějším z hlediska časového a výpočetního.

Tabulka 5.30: Metrika přesnosti klasifikátorů

Klasifikátor	přesnost
LSTM	61 %
GRU	61 %
LSTM s technikou GloVe	61 %
GRU s technikou GloVe	60 %
LSTM s technikou Attention	60 %
GRU s technikou Attention	**63 %**
BERT	**63 %**

Kapitola 6

Závěr diplomové práci

V této diplomové práci jsem se snažil vytvořit strojově naučený model, který by byl schopný detekovat strojově přeložené texty. Provedl jsem řadu experimentů s různými architekturami a úpravami, abych našel optimální modely, které by s využitím znalostí z oblasti zpracování přirozeného jazyka a sentimentální analýzy dokázaly rozpoznávat rozdíly mezi větami. Navzdory nízké úspěšnosti považuji modely za potenciálně schopné řešit tento problém.

Hlavní překážkou byl nedostatek dostupných dat. Vytvoření dostatečně velkého datasetu představovalo výzvu vzhledem k potřebě shromáždit rozsáhlý textový korpus a provést jeho překlad pomocí nástrojů, které nejsou primárně určeny pro překlad velkých textových segmentů, ale spíše se zaměřují na překlad jednotlivých vět nebo odstavců. Tento proces by vyžadoval značné zdroje a prostředky. Bohužel, z důvodu těchto omezení nebylo možné vytvořit dostatečný dataset pro účely této studie.

6.1 Možná vylepšení

Jednou z hlavních možností vylepšení je rozšíření textového datového souboru, a to významným způsobem. Důkladné rozšíření datového souboru by mohlo přispět k zlepšení výkonu modelu. Dále na základě rozšířeného datového souboru by bylo vhodné průběžně upravovat parametry architektur s cílem dosáhnout lepších výsledků. Pravidelné úpravy parametrů a experimentování s různými nastaveními mohou vést ke zdokonalení modelů.

6.2 Zkušenost

Díky této práci jsem získal neuvěřitelné množství znalostí, a to od samého začátku. Zjistil jsem, jak je důležité mít k dispozici skvělý dataset, ze kterého lze vypočítat požadované výsledky. Dále jsem získal zkušenost s vytvářením vlastních modelů a také s technikou fine-tuningu před-trénovaných modelů.

Nabyté zkušenosti mi umožnily porovnávat různé modely a hodnotit je na základě specifických kritérií. Dále jsem získal schopnost detekovat nedostatky a problémy v modelech, pokud se objeví. Všechny tyto dovednosti a znalosti jsem získal prostřednictvím této práce.

Chtěl bych tedy znovu vyjádřit své upřímné poděkování panu vedoucímu práce, panu Platošovi, za poskytnutí této příležitosti pracovat na této úžasné oblasti a za jeho cennou podporu během celého mého výzkumu.

Literatura

1. *Natural language processing* [online]. [cit. 2021-04-25]. Dostupné z: https://en.wikipedia.org/wiki/Natural_language_processing.
2. *Zpracování přirozeného jazyka* [online]. [cit. 2021-04-25]. Dostupné z: https://cs.wikipedia.org/wiki/Zpracov%C3%A1n%C3%AD_p%C5%99irozen%C3%A9ho_jazyka.
3. VAIDYA, Neeraja. *5 Natural Language Processing Techniques for Extracting Information* [online]. [cit. 2021-04-25]. Dostupné z: https://blog.aureusanalytics.com/blog/5-natural-language-processing-techniques-for-extracting-information.
4. *Word embeddings* [online]. [cit. 2021-04-25]. Dostupné z: https://www.tensorflow.org/tutorials/text/word_embeddings.
5. *A Beginner's Guide to Word2Vec and Neural Word Embeddings* [online]. [cit. 2021-04-25]. Dostupné z: https://wiki.pathmind.com/word2vec.
6. PENNINGTON, Jeffrey; SOCHER, Richard; MANNING, Christopher. GloVe: Global Vectors for Word Representation. In: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Doha, Qatar: Association for Computational Linguistics, 2014-10, s. 1532–1543. Dostupné z DOI: 10.3115/v1/D14-1162.
7. PHD, Jason Brownlee. *A Gentle Introduction to the Bag-of-Words Model* [online]. [cit. 2021-04-25]. Dostupné z: https://machinelearningmastery.com/gentle-introduction-bag-words-model/.
8. DOSHI, Sanket. *Skip-Gram: NLP context words prediction algorithm* [online]. [cit. 2021-04-25]. Dostupné z: https://towardsdatascience.com/skip-gram-nlp-context-words-prediction-algorithm-5bbf34f84e0c.
9. *Umělá neuronová síť* [online]. [cit. 2021-04-25]. Dostupné z: https://cs.wikipedia.org/wiki/Um%C4%9Bl%C3%A1_neuronov%C3%A1_s%C3%AD%C5%A54.
10. HARRISON KINSLEY, Daniel Kukieła. *Neural Networks from Scratch in Python*. 2020.
11. KOSTADINOV, Simeon. *How Recurrent Neural Networks work* [online]. [cit. 2021-04-25]. Dostupné z: https://towardsdatascience.com/learn-how-recurrent-neural-networks-work-84e975feaaf7.

12. PEDAMALLU, Hemanth. *RNN vs GRU vs LSTM* [online]. [cit. 2021-04-25]. Dostupné z: https://medium.com/analytics-vidhya/rnn-vs-gru-vs-lstm-863b0b7b1573.

13. *Understanding LSTM Networks* [online]. [cit. 2021-04-25]. Dostupné z: http://colah.github.io/posts/2015-08-Understanding-LSTMs/.

14. KOSTADINOV, Simeon. *Understanding GRU Networks* [online]. [cit. 2021-04-25]. Dostupné z: https://towardsdatascience.com/understanding-gru-networks-2ef37df6c9be.

15. *Google Brain* [online]. [cit. 2022-05-27]. Dostupné z: https://en.wikipedia.org/wiki/Google_Brain.

16. VASWANI, Ashish; SHAZEER, Noam; PARMAR, Niki; USZKOREIT, Jakob; JONES, Llion; GOMEZ, Aidan N.; KAISER, Lukasz; POLOSUKHIN, Illia. Attention is All you Need. In: GUYON, I.; LUXBURG, U. Von; BENGIO, S.; WALLACH, H.; FERGUS, R.; VISHWANATHAN, S.; GARNETT, R. (ed.). *Advances in Neural Information Processing Systems*. Curran Associates, Inc., 2017, sv. 30. Dostupné také z: https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.

17. *Transformer (machine learning model)* [online]. [cit. 2022-05-27]. Dostupné z: https://en.wikipedia.org/wiki/Transformer_(machine_learning_model).

18. *Transformer model for language understanding* [online]. [cit. 2022-05-29]. Dostupné z: https://www.tensorflow.org/text/tutorials/transformer.

19. *Python* [online]. [cit. 2023-04-20]. Dostupné z: https://www.python.org/.

20. DEVLIN, Jacob; CHANG, Ming-Wei; LEE, Kenton; TOUTANOVA, Kristina. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. Minneapolis, Minnesota: Association for Computational Linguistics, 2019-06, s. 4171–4186. Dostupné z DOI: 10.18653/v1/N19-1423.

21. JACOB DEVLIN, Ming-Wei Chang. *Open Sourcing BERT: State-of-the-Art Pre-training for Natural Language Processing* [online]. [cit. 2022-05-28]. Dostupné z: https://ai.googleblog.com/2018/11/open-sourcing-bert-state-of-art-pre.html.

22. *Google Colab* [online]. [cit. 2022-05-27]. Dostupné z: https://colab.research.google.com/.

23. *European Union Web* [online]. [cit. 2023-04-20]. Dostupné z: https://european-union.europa.eu.

24. *Deepl* [online]. [cit. 2023-04-20]. Dostupné z: https://www.deepl.com.

25. *Google Translate* [online]. [cit. 2023-04-20]. Dostupné z: https://translate.google.com/.