VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

AUTOMATICKÉ HODNOCENÍ HUMORU

AUTOMATIC HUMOR EVALUATION

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE **JOSEF KATRŇÁK**
AUTHOR

VEDOUCÍ PRÁCE **Ing. MARTIN DOČEKAL**
SUPERVISOR

BRNO 2021

Vysoké učení technické v Brně
Fakulta informačních technologií

Ústav počítačové grafiky a multimédií (UPGM) Akademický rok 2020/2021

Zadání bakalářské práce

Student: **Katrňák Josef**
Program: Informační technologie
Název: **Automatické hodnocení humoru**
Automatic Humor Evaluation
Kategorie: Zpracování řeči a přirozeného jazyka

Zadání:

1. Nastudujte metody hodnocení humoru pomocí strojového učení v textových datech.
2. Navrhněte systém schopný hodnotit humor v textu.
3. Implementujte zmíněný systém.
4. Otestujte systém na vybraných datových sadách.
5. Zhodnoťte dosažené výsledky pomocí standardních metrik.

Literatura:

- dle doporučení vedoucího

Podrobné závazné pokyny pro vypracování práce viz https://www.fit.vut.cz/study/theses/

Vedoucí práce: **Dočekal Martin, Ing.**
Vedoucí ústavu: Černocký Jan, doc. Dr. Ing.
Datum zadání: 1. listopadu 2020
Datum odevzdání: 12. května 2021
Datum schválení: 30. října 2020

Abstrakt

Cílem této práce je vytvoření systému pro automatické hodnocení humoru. Systém umožňuje predikovat vtipnost a kategorii pro vstup zadaný v angličtině. Hlavní podstatou je vytvoření klasifikátoru a trénování modelu na vytvořených datových sadách pro získání co nejlepších výsledků. Architektura klasifikátoru je založena na neuronových sítích. Systém zároveň obsahuje webové uživatelské rozhraní pro komunikaci s uživatelem. Výsledek je webová aplikace propojená s klasifikátorem umožňující hodnocení uživatelského vstupu a poskytování zpětné vazby od uživatelů.

Abstract

The aim of this thesis is to create a system for automatic humor evaluation. The system allow to predict humor and category for english input. The main essence is to create a classifier and train the model with the created datasets to get the best possible results. The classifier architecture is based on neural networks. The system also includes a web user interface for communication with the user. The result is a web application linked to a classifier that allows user input to be evaluated and user feedback to be provided.

Klíčová slova

hodnocení humoru, klasifikace, zpracování přirozeného jazyka, hluboké učení, neuronové sítě, transformer, BERT

Keywords

humor evaluation, classification, natural language processing, deep learning, neural networks, transformer, BERT

Citace

KATRŇÁK, Josef. *Automatické hodnocení humoru*. Brno, 2021. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Martin Dočekal

Automatické hodnocení humoru

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Ing. Martina Dočekala. Uvedl jsem všechny literární prameny, publikace a další zdroje ze kterých jsem čerpal.

.....
Josef Katrňák
10. května 2021

Poděkování

Děkuji panu Ing. Martinovi Dočekalovi za vedení a pomoc při tvorbě této práce.

Obsah

Seznam obrázků

Kapitola 1

Úvod

Humor je nedílnou a každodenní součástí mezilidské interakce. I když existují tři základní filozofické teorie humoru, stanovení přesné definice humoru je problematické a neexistuje jednotná a všeobecně uznávaná teorie. I kvůli tomuto problému v oblasti filozofie není počítačové hodnocení humoru jednoduchým úkolem. Porozumění humoru počítačem by přineslo potencionální benefity použitelné pro interakci člověka a počítače. A právě automatickým hodnocením humoru se zabývá tato práce. Vytvoření systému pro automatické hodnocení humoru by mohlo být velice prospěšné pro tvůrce obsahu, kterým by takový systém poskytl nástroj k ověřování jejich textů.

Díky rozvoji v odvětví zpracování přirozeného jazyka je klasifikace textových dat čím dál více efektivnější. Nejmodernější technologie jsou schopny s velkou přesností predikovat vstupní data do správných tříd. Práce vytváří systém, který popisuje a využívá tyto technologie pro klasifikaci vtipnosti a kategorie vtipu.

Cílem práce je vytvoření systému skládajícího se z klasifikátoru a uživatelského webového prostředí. Klasifikátor určuje vtipnost a kategorii vstupu. Webová aplikace slouží k prezentování výsledků klasifikátoru a k získání případné uživatelské zpětné vazby. Celý systém komunikuje s uživatelem v angličtině. Model pro klasifikaci, který je trénován na vytvořených datových sadách, umožňuje klasifikovat tři úrovně vtipnosti a šestnáct kategorií vtipů.

Kapitola 2 se zaměřuje na teoretickou stránku humoru a na existující řešení automatického hodnocení humoru. Kapitola 3 se podrobněji věnuje oboru zpracování přirozeného jazyka a využití hlubokého učení pro klasifikaci textu. Návrh systému je popsán v kapitole 4 a jeho implementace v kapitole 5. Kapitola 6 se věnuje datovým sadám, a to existujícím i těm v práci vytvořeným. Samotné trénování, ke kterému se využívají vzniklé datové sady, a další experimenty jsou popsány v kapitole 7. Poslední kapitola 8 prezentuje konečné zhodnocení projektu a možnosti budoucího rozšíření.

Kapitola 2

Teorie humoru

Humor je součástí každodenního života většiny lidí. Ve vztazích je smysl pro humor páry ceněn jako důležitá vlastnost [86]. I přes to ale od filozofů neexistuje mnoho textů na toto téma. A existující texty jsou většinou k humoru kritické [65, 3, 18].

Od starověkého Řecka do 20. století se většina filozofických textů zaměřovala na spíše pohrdavý a výsměšný typ humoru. Platón považoval smích za emoci potlačující racionální sebekontrolu [65] a za zlomyslný [64]. Aristotelés také považoval smích za opovrženíhodný, i když považoval vtip za cennou část konverzace [3]. Stoikové souhlasili s Platónem, že smích snižuje sebekontrolu [18]. Negativní pohled na humor pokračoval i u raných křesťanských myslitelů a přes ně se přenesl na pozdější evropskou kulturu. V Bibli je humor reprezentován negativně a většinou souvisí s nepřátelstvím. Křesťanské evropské odmítání humoru pokračovalo i ve středověku. Puritáni silně odsuzovali humor, psali traktáty proti smíchu a komedii, a dokonce zakázali komedie, když se v 17. století dostali k vládě [55].

V této době se proti humoru ohrazovali Thomas Hobbes a René Descartes. Thomas Hobbes považoval lidské bytosti za individualistické a soutěživé. Hobbes uvedl, že pokud si lidská bytost rychle uvědomí znamení, že je nadřazená, dobré pocity planoucí z tohoto znamení pravděpodobně vyústí ve smích [25]. Descartes objevil podobné vysvětlení pro smích. I když připustil možné jiné důvody smíchu, považoval smích pouze za výraz pohrdání a výsměchu [14]. Z výroků Hobbese a Descartese se zrodila teorie superiority [55].

2.1 Teorie superiority

Teorie superiority (převahy, nadřazenosti) je založena na tom, že náš smích vyjadřuje pocit nadřazenosti nad jinými jedinci nebo nad naším dřívějším stavem. Zastáncem této teorie je Roger Scruton. Roger Scruton prohlásil, že pokud se lidem nelíbí, že se jim někdo směje, tak je to kvůli tomu, že smích v očích zesměšňujícího znehodnocuje cíl posměchu [54].

V 18. století dominance teorie superiority začíná polevovat. Francis Hutchenson napsal kritiku na Hobbesovu úvahu o smíchu. Hutchenson argumentuje, že pocity nadřazenosti nejsou ani nezbytné, ani dostačující pro smích a při smíchu se nemusíme porovnávat s ostatními [29].

Smích v nás zpravidla nevyvolává pocit nadřazenosti oproti zvířatům, ani pocit lítosti, který může souviset s nadřazeností. Naopak nás můžou rozesmát komické postavy předvádějící překvapivé dovednosti, které nám samotným chybí. Takový smích nevyžaduje porovnání s danou postavou a i kdybychom se sní porovnali, tak se nebudeme považovat za nadřa-

zené. Někdy se lidé smějí i sami sobě (ne svému dřívějšímu stavu), což je v rozporu s teorií superiority. Existují i další případy smíchu, kde se nevyskytuje osobní porovnávání.

Další oslabování dominance teorie superiority vyústilo v dvě nové teorie humoru. Jedná se o teorii relaxace a teorii inkongruence. Ani jedna z těchto teorií dále nezmiňuje pocity nadřazenosti [55].

2.2 Teorie relaxace

Teorie relaxace vysvětluje smích v nervové soustavě jako přetlakový ventil v parním kotli [55]. Teorie se začala formovat v roce 1709, publikací Lorda Shaftesburyho *An Essay on the Freedom of Wit and Humor* [76]. Vědci v této době věděli, že nervy spojují mozek se smyslovými orgány a svaly, ale mysleli si, že nervy nesou takzvané *zvířecí duchy*, což jsou plyny a kapaliny jako například vzduch a krev. Shaftesbury vysvětluje smích jako uvolnění těchto zvířecích duchů tvořících tlak uvnitř nervů.

V následujících dvou stoletích přišel pokrok v porozumění nervovému systému. Myslitelé jako Herbert Spencer nebo Sigmund Freud revidovali původní biologické vysvětlení teorie relaxace, ale zachovali myšlenku, že smích uvolňuje zadrženou nervovou energii. Freud považoval smích za uvolnění nadbytečné nervové energie, která byla v jedinci nahromaděna pro určitý psychologický úkol, který ale nakonec daný jedinec neuskutečnil [23].

Víme, že existuje spojení mezi smíchem a vydanou energií. I přesto ale jen málo odborníků zastává teorii relaxace. Existují situace, kde smích zřejmě neuvolňuje žádnou nahromaděnou energii. Příkladem mohou být krátké vtipy dlouhé jen pár slov. Samotný základ teorie relaxace, *hydraulický model* nervového systému, je dnes již zastaralý [55].

2.3 Teorie inkongruence

Teorie inkongruence říká, že smích je způsoben zpozorováním nesouladu, tedy něčeho co naruší naše mentální vzory a předpoklady. Tato teorie byla podpořena mysliteli, jako jsou James Beattie, Immanuel Kant nebo Arthur Schopenhauer. Dnes se u komiků používají techniky *set-up* a *punchline*, kde set-up je první částí vtipu, která vytváří očekávání a punchline nakonec toto očekávání rozbíjí. Konec vtipu je tedy rozporný se začátkem. Ve filosofii a psychologii je nyní považována za dominantní teorii humoru [55].

Již Aristoteles a Cicero uvažovali, že pro vyvolání smíchu u publika je třeba vytvořit očekávání a následně výsledek tohoto očekávání porušit. První filozof, který zmínil inkongruenci byl James Beattie. Beattie považuje za důvod smíchu dvě nebo více rozporuplných částí určitého celku [5]. Immanuel Kant nachází původ smíchu v náhlé transformaci napjatého očekávání v nic [34]. Arthur Schopenhauer nachází nesoulad vedoucí k smíchu mezi naším smyslovým vnímáním věci a naší abstraktní racionální znalostí stejných věcí [75], jedná se tedy o rozpor mezi abstraktními myšlenkami a skutečnými věcmi.

Základem teorie inkongruence je tedy nějaká zpozorovaná věc nebo událost, která narušuje naše obvyklé mentální vzory a normální očekávání. Různí myslitelé přidávají k teorii rozdílné detaily. Thomas Schultz a Jerry Suls prohlásili, že si lidé humor neužívají díky samotnému rozporu, ale díky rozluštění rozporu. Pobavení podle této teorie je podobné jako u řešení hádanek. Jiní teoretici vyvrací tyto detaily a tvrdí, že řešení rozporu se vztahuje jen na část humoru [55].

Rozpor samotný ale není dostatečný pro humor. Nesoulad může vyvolat strach nebo vztek. Michael Clark předpokládá, že pobavení není pouze reakce na rozpor, ale způsob

užívání si rozporu [54]. Ale ani toto vysvětlení nemusí být dostatečně specifické, jelikož užívání si nesouladu může vyústit i v nehumorný požitek.

2.4 Další teorie

Kromě klasických teorií existují i další teorie, které zakládají na teorii inkongruence. **Sémanticko-skriptová teorie humoru** (anglicky Script Semantic Theory of Humour) byla představena Viktorem Raskinem. Teorie tvrdí, že každý text je spojen se sémantickým skriptem[1] pomocí nějaké kognitivní struktury člověka, která poté umožňuje ohodnotit text jako humorný. Zavádí se dvě podmínky pro humorný text. První podmínka říká, že text je plně nebo částečně kompatibilní s dvěma rozdílnými sémantickými skripty. Druhá podmínka požaduje, aby byly tyto dva skripty protikladné a plně nebo částečně se překrývaly. Humor se objeví na konci vtipu, kde publikum náhle změní své chápání z primárního skriptu na sekundární (protikladný) skript [71].

Všeobecná teorie verbálního humoru (anglicky General Verbal Theory of Humour) si zakládá na šesti nezávislých úrovních *zdrojů vědomostí*, které představili Viktor Raskin a Salvator Attardo [4]. Tyto zdroje vědomostí se mohou použít pro modelování jednotlivých vtipů nebo pro analýzu podobnosti mezi vtipy. Mezi tyto zdroje vědomostí patří:

- **Skriptová opozice** - představena v Sémanticko-skriptové teorii humoru
- **Logický mechanismus** - mechanismus propojující rozdílné skripty vtipu
- **Situace** - je charakterizována objekty vtipu
- **Cíl** - terč vtipu, na koho vtip útočí
- **Narativní strategie** - narativní formát vtipu
- **Jazyk** - informace pro verbalizaci textu

Díky zdrojům vědomostí tato teorie obsáhleji zabírá teorii humoru, než dříve zmíněná Sémanticko-skriptová teorie. I přesto má ale teorie nedostatky. Teorie nevysvětluje například situace, kde je humor vyvolán emočním nesouladem bez ohledu na situaci, logický mechanismus či skriptovou opozici.

2.5 Shrnutí teorií humoru

Teorie superiority předpokládá, že pobavení vychází z pocitu nadřazenosti. Teorie relaxace zase považuje smích za uvolnění přebytečné energie. Dominantní teorie inkongruence hledá důvod smíchu v rozporu mezi očekávaným a skutečným.

I přes vyžadované upřesnění se zdá, že teorie inkongruence humor a smích vysvětluje lépe než vědecky zastaralá teorie relaxace. Zároveň je teorie inkongruence obsáhlejší než teorie superiority, jelikož bere v potaz i jiné druhy humoru, než pouze ty založené na nadřazenosti. Příkladem jiného druhu humoru mohou být slovní hříčky.

I když všechny tři teorie nabízejí užitečný náhled na původ smíchu, žádná z těchto teorií nedokáže plně vysvětlit humor jako celek. Humor má spoustu druhů, forem a kontextuálních závislostí, které není jednoduché pojmout obecnou definicí.

[1] Skript je organizovaný souhrn informací o něčem. Jedná se o kognitivní strukturu poskytující informace o věcech.

2.6 Automatické hodnocení humoru

Humor je důležitým prvkem komunikace. Kromě poskytování zábavy pomáhá k regulování konverzací a hraje významnou roli v budování mezilidských vztahů [7]. Právě proto by měly být počítačové systémy schopné s humorem pracovat. V interakci mezi počítačem a člověkem může mít zapojení humoru pozitivní vliv na psychický stav zúčastněných lidí [57]. Existují práce zabývající se generováním [80, 39, 89] i detekcí [47, 85, 8] humoru.

Detekce, hodnocení a generování humoru je stále jedním z náročných problémů v oboru umělé inteligence. Humor totiž vyžaduje všeobecné znalosti a schopnost vnímat vztahy mezi entitami a objekty na různých úrovních porozumění jazyka. Zároveň je třeba počítat s tím, že smysl pro humor se liší člověk od člověka.

Většina prací zaměřující se na automatickou detekci humoru vychází z teorie inkongruence. Pomocí inkongruence je možné stanovit pomyslnou hranici mezi vtipným a nevtipným textem, ale je těžké stanovit co přesně dělá text vtipným. Teorie stavící na inkongruenci jako Sémanticko-skriptová teorie humoru a Všeobecná teorie verbálního humoru se tohoto snaží dosáhnout.

Existující přístupy

Na problém hodnocení humoru se lze dívat jako na tradiční klasifikační úkol, kde se klasifikátoru poskytnou pozitivní (vtipné) a negativní (nevtipné) záznamy. *Mihalcea a Strapparava* [47] se zaměřují právě na tuto klasifikaci skrze experimenty na velkých datových sadách. Práce se zabývá výhradně krátkými vtipy (tzv. *one-liners*) a binární klasifikaci vtipu založenou na stylistických a obsahových prvcích. Použity jsou textové klasifikátory využívající metodu Naive Bayes a metodu podpůrných vektorů (anglicky support vector machines, SVM). Naive Bayes klasifikátor je jednoduchý pravděpodobnostní klasifikátor založený na Bayesově větě, který předpokládá slovní nezávislost.

Ahuja, Bali a Singh [1] se zaměřují na klasifikaci humoru pomocí modus operandi, tématu a kategorie vtipu. **Modus operandi** určuje způsob jakým je vtip podán potenciálnímu publiku (například jestli je vtip sarkastický). **Téma** se zaměřuje na obsahové a jazykové aspekty vtipu (rozlišuje tedy například černý humor). **Kategorie** znázorňuje klíčový koncept vtipu (například jestli se jedná o vtip o zvířatech). Mezi technologie využité v této práci patří lineární diskriminační analýza (anglicky linear discriminant analysis, LDA), Naive Bayes, metoda podpůrných vektorů (anglicky support vector machines, SVM) a logistická regrese (LR).

S rostoucí popularitou neuronových sítí se tato technologie začala používat i pro hodnocení humoru. *Chen a Soo* [8] pro tyto účely implementují konvoluční neuronovou síť (anglicky convolutional neural network, CNN). Neuronové sítě dovolují pomocí datových sad trénovat model bez lidské interakce. Práce kromě anglických záznamů pracuje i s daty v čínštině.

Wilbur a Campbell [85] využívají architekturu Transformer k detekci, zda je text vtipný nebo nevtipný. K této detekci je použit již před-trénovaný model BERT, který se následně doladil pro klasifikaci humoru.

Kapitola 3

Zpracování přirozeného jazyka

Zpracování přirozeného jazyka (anglicky Natural Language Processing, NLP) je počítačový přístup k analýze textu. Zabývá se lidským porozuměním a použitím jazyka k vytvoření odpovídajících nástrojů a technik, které dovolí počítačům porozumět přirozené řeči a manipulovat s ní. Lidé sepisují různé věci už po tisíce let, a proto je zde snaha naučit se číst a rozumět všem těmto datům. Počítače sice zatím nerozumí přirozenému jazyku tak jako lidé, ale díky NLP toho dokážou již mnoho. Aplikace NLP může ušetřit čas ve spoustě případů.

NLP lze definovat jako teoreticky motivovaný rozsah počítačových technik pro analýzu a prezentaci přirozeně se vyskytujících textů na jedné, či více úrovních lingvistické analýzy, za účelem dosažení člověku podobnému zpracování jazyka pro řadu úkolů a použití [41].

Rozsah počítačových technik říká, že existuje více metod a technik, které lze použít pro dosažení určitého typu jazykové analýzy.

Počítače jsou skvělé pro práci se strukturovanými daty, mezi které patří například tabulky a databáze. Bohužel pro počítače, lidé nekomunikují tabulkami ale pomocí slov v přirozeně se vyskytujících textech. Mezi *přirozeně se vyskytující texty* patří sepsané texty v jakémkoliv jazyce. Požadavek přirozeně se vyskytujícího textu je, že se musí jednat o text použitý v komunikaci mezi lidmi. Text by tedy neměl být upraven pro analýzu, naopak by měl být získán ze skutečné mezilidské komunikace. Taková data jsou příkladem nestrukturovaných dat. Nestrukturovaná data představují většinu dostupných údajů v reálném světě. Přibližně 80 až 90 procent informací ve většině organizací jsou nestrukturovaná data [69].

Existují různé druhy jazykového zpracování, které jsou popsány pomocí *úrovní lingvistické analýzy*. Předpokládá se, že člověk při produkci nebo příjmu jazyka využívá obvykle všechny úrovně. Každá úroveň vyjadřuje určitý typ významu. NLP systémy se odlišují tím jakou úroveň nebo kombinaci úrovní využívají.

Cílem NLP je dosažení *člověku podobnému zpracování jazyka*. To znamená, že NLP systém by měl být schopný pracovat s textem na podobné úrovni jako člověk. Cíle zahrnují schopnosti parafrázovat vstupní text, přeložit vstupní text, odpovědět na otázku z textu a vyvozovat závěry z textu.

V praxi se NLP používá pro splnění *řady úkolů a použití*. Samotné zpracování přirozeného jazyka tedy vetšinou není cílem, ale prostředkem k cílovému použití. Příkladem použití je získávání informací (anglicky Information Retrieval, IR), strojový překlad (anglicky Machine Translation, MT) nebo třeba odpovídání na otázky (anglicky Question Answering, QA).

3.1 Obory související se zpracováním přirozeného jazyka

Jelikož se NLP snaží o zpracování jazyka, které je výkonnostně na podobné úrovni jako zpracování lidské, je vhodné ho členit do oboru umělé inteligence (anglicky Artificial Intelligence, AI). NLP se prolíná mnoha obory, mezi primární patří lingvistika, počítačová věda a kognitivní psychologie [41].

Umělá inteligence

Umělá inteligence je obor, který se zabývá vytvářením systémů s charakteristickým chováním, které lze asociovat s inteligencí lidského chování. Příkladem takového chování je vnímání, řešení problémů, plánování, učení, adaptace, vliv na prostředí a právě zpracování přirozeného jazyka [81].

Lingvistika

Lingvistika je obor zabývající se výzkumem jazyka. V základu lze jazyky rozdělit do dvou skupin:

- jazyky **přirozené**
- jazyky **umělé**

Zatímco umělé jazyky jsou záměrně vytvořené lidmi, přirozené jazyky vznikly spontánně a jsou primárně používány pro komunikaci mezi lidmi. Přirozené jazyky se stále vyvíjejí. Mezi přirozené jazyky patří i čeština a angličtina [43].

Pro NLP je lingvistika důležitá, protože zkoumá vlastní strukturu jazyka, konkrétně z jakých prvků se slova skládají nebo jak se slova kombinují do vět [58].

Počítačová věda

Počítačová věda (matematická informatika, anglicky computer science) studuje procesy interagující s daty, a to z hlediska hardware i software [6]. Zabývá se vytvářením interní reprezentace dat a následným efektivním zpracováním daných struktur.

Kognitivní psychologie

Kognitivní psychologie je věda studující vědomí, duševní procesy podmiňující chování, včetně myšlení, usuzování, rozhodování a v určité míře i motivaci a emoce [19]. Cílem je modelovat použití jazyka psychologicky věrohodným způsobem.

3.2 Hluboké učení

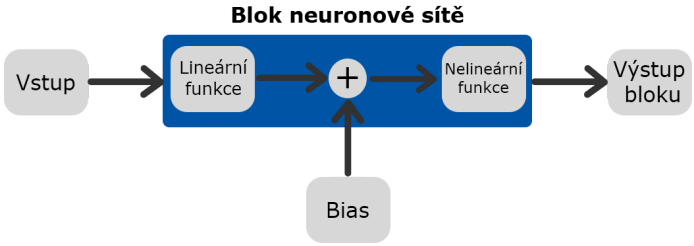
Hluboké učení (anglicky deep learning) je disciplína spadající pod strojové učení (anglicky machine learning), která se zabývá především využitím neuronových sítí. Neuronové sítě (anglicky neural networks, NN) se dnes používají ve spoustě odvětví. NN pomáhají například při převodu textu na řeč [24], autonomním řízení aut [9], rozpoznávání tváří [13] nebo právě zpracování přirozeného jazyka 3.3.

Vývoj neuronových sítí začal stavět na chování lidských neuronů [21]. I když začátky byly pro neuronové sítě těžké, díky současnému velkému množství dat a výkonu grafických

procesorů jsou dnes neuronové sítě velmi populární, a téměř každá výzkumná oblast je jimi ovlivněna [77].

Struktura neuronových sítí

Základní struktura jednoho bloku neuronové sítě, která je zobrazena na obrázku 3.1, obsahuje **lineární funkci** následovanou **nelineární funkcí**. Každý blok neuronové sítě obsahuje **sadu parametrů** (váhy a bias), které se aktualizují během trénování. Cílem je minimalizovat předdefinovanou **ztrátovou funkci** [77].



Obrázek 3.1: Jeden blok neuronové sítě obsahující lineární a nelineární elementy.

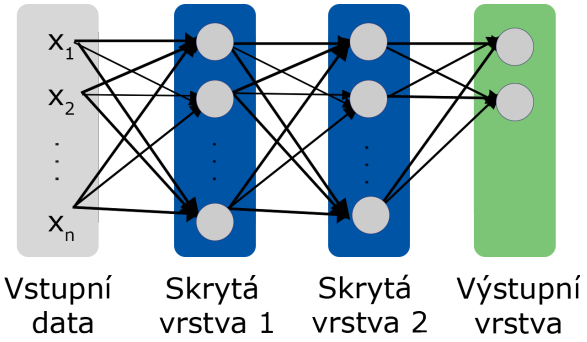
Pokud je vektor vstupních dat označen x, matice vah (lineární funkce) W, vektor pro bias b a nelineární funkce Ψ, pak se výstupní vektor y vypočítá jako 3.1.

$$y = \Psi(Wx + b) \tag{3.1}$$

Bloky se mohou skládat za sebe, čímž vzniká vrstvená struktura neuronové sítě. Vstupní data procházejí sítí a postupně se zpracovávají. Tomuto procesu se anglicky říká **forward pass**. Během trénování se běžně používá proces zpětného šíření (anglicky **backpropagation**), který umožňuje aktualizaci parametrů sítě.

Vícevrstvý percepton a konvoluční neuronové sítě

Často je blok neuronové sítě realizován jako **plně propojená** (anglicky Fully-connected, FC) vrstva. Síť, která vznikne spojením FC vrstev, se nazývá **vícevrstvý percepton** (anglicky Multi-Layer Percepton, MLP) [73]. FC vrstva spojuje každý neuron jedné vrstvy s každým neuronem následující vrstvy, jak je zobrazeno na obrázku 3.2. Díky tomu se informace šíří ze všech neuronů jedné vrstvy do všech neuronů následující vrstvy.



Obrázek 3.2: Plně propojené vrstvy.

Využitím **konvoluční vrstvy** (anglicky convolution layer), vznikají **konvoluční neuronové sítě** (anglicky convolutional neural network, CNN). Tato vrstva aplikuje na vstup jeden nebo více konvolučních filtrů. Konvoluční operace udržují prostorové informace, a proto jsou často používány pro zpracování obrazu.

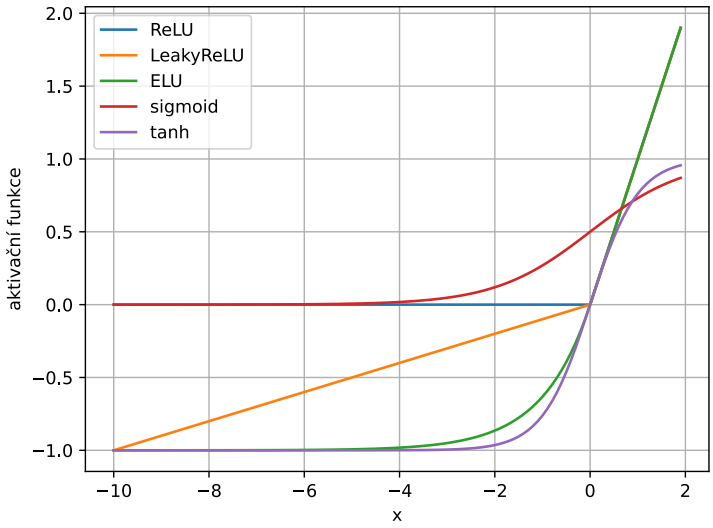
Rekurentní neuronové sítě

Dalším zástupcem neuronových sítí jsou **rekurentní neuronové sítě** (anglicky Recurrent Neural Network, RNN) [45]. RNN síť přijímá v každém kroku pouze jeden nový vstup a zpětnou vazbu, která je vypočítána pomocí výsledku stejné sítě z předešlého kroku. Díky tomu si síť pamatuje informace z předchozích kroků a teoreticky neklade omezení na délku vstupu. Komplexnější RNN struktury zpracovávají vstup z obou směrů nebo přidávají hradla ke zpětné vazbě a vstupu sítě.

Nejznámější komplexní architektura, která přidává hradla k RNN, je **Long-Short-Term-Memory** (LSTM) [26]. Hradla rozhodují jaká informace se použije pro výpočet výstupu a zpětné vazby, a jaká informace se vymaskuje (zapomene). Díky tomu se zjednodušuje kombinace předchozích a současných informací.

Aktivační funkce

Nelineární funkce definované pro každou vrstvu dovolují modelovat nelineární závislosti. Pokud jsou aplikovány po jednotlivých prvcích, pak se nazývají **aktivační funkce** (anglicky activation functions). Aktivační funkce bere výstup neuronu a převádí jej do formy vhodné pro vstup dalšího neuronu. Mezi nejznámější aktivační funkce patří Rectified Linear Unit (ReLU) [12], leaky ReLU [88], Exponential Linear Unit (ELU) [11], sigmoid a hyperbolický tangens (tanh). Porovnání průběhů funkcí je zobrazeno na obrázku 3.3.



Obrázek 3.3: Porovnání aktivačních funkcí, kde pro leaky ReLU je $\alpha = 0.1$ a pro ELU je $\alpha = 1$.

Pooling

Další často používaná nelineární operace v NN modelech je pooling funkce. Jedná se o agregační operaci, která redukuje velikost, ale snaží se zachovat dominantní vlastnosti.

Mezi nejznámější pooling funkce patří **max pooling**, která vybírá maximální hodnoty, a **mean pooling**, která vybírá průměrné hodnoty z daných oblastí.

Softmax

Softmax funkce normalizuje vektory na rozdělení pravděpodobnosti nad předpokládanými třídami výstupu. Vstupem softmax funkce jsou **logits**. Logits představují výstup neuronové sítě transformovaný do velikosti $1 \times N$, kde N je počet tříd. Pokud je logits označení v, pak se softmax vypočítá jako 3.2.

$$\operatorname{softmax}(v)_i = \frac{\mathrm{e}^{v_i}}{\sum_{j=1}^{N} \mathrm{e}^{v_j}}, \quad i \in [1, \ldots, N] \tag{3.2}$$

Ztrátová funkce

Ztrátová funkce (anglicky loss function) se zpravidla volí na základě datové sady a daného úkolu. Pro klasifikaci, kde je cílem identifikace správné třídy (případně správných tříd) se často jako ztrátová funkce volí **křížová entropie** (anglicky cross-entropy).

Implementace ztrátové funkce pomocí křížové entropie je založena na normalizovaném vektoru pravděpodobností a odpovídajícím seznamu potencionálních výsledků. Normalizovaný vektor se vypočítá pomocí softmax funkce 3.2. Pokud y_i je reálná pravděpodobnost toho, že vstup patří do třídy i a p_i je predikce modelu pro tuto třídu, pak se ztrátová funkce křížové entropie $\mathcal{L}$ vypočítá jako 3.3. y_i je většinou binární a obsahuje 1 na jediném indexu, který odpovídá správné třídě (tzv. *one-hot encoding*).

$$\mathcal{L} = -\sum_{i=1}^{N} y_i \log(p_i) \tag{3.3}$$

Trénování neuronové sítě

Každý neuron používá k výpočtu sadu parametrů. Trénování spočívá v iterativním ladění těchto parametrů. Cílem je aktualizovat váhy tak, aby byla ztrátová funkce pro danou trénovací sadu co nejmenší. K efektivnímu trénování neuronové sítě je obvykle potřeba velká datová sada a GPU.

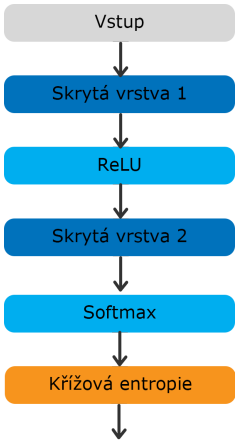
Trénovací metody se dělí na trénování **s učitelem** (anglicky supervised) a **bez učitele** (anglicky unsupervised). Trénování s učitelem oproti trénování bez učitele vyžaduje anotovaná data, ale obecně dosahuje lepších výsledků.

Minimalizace předdefinované ztrátové funkce probíhá ve dvou fázích. Nejprve se provede forward pass vstupních dat přes všechny vrstvy sítě a z predikovaných tříd a reálných dat se vypočte chyba. Následně se provede zpětné šíření chyb od poslední k první vrstvě a aktualizují se váhy. Tento proces probíhá nepřetržitě ve snaze najít nejoptimálnější hodnoty pro váhy sítě.

Backpropagation poskytuje gradienty, které se používají k aktualizaci vah sítě. Výpočet gradientů je výpočetně náročný, a proto se aplikuje pouze na podmnožiny vstupních dat, kterým se říká (mini-)**batches**. Rozdělení dat do batches také přispívá k urychlení

trénování, protože aktualizace parametrů se provádí po každé batch a ne pouze jednou na všech datech naráz. Výpočet na celé datové sadě se tedy provádí ve více cyklech. Cyklus přes všechna data datové sady se nazývá **epocha**. Pro každou epochu se data náhodně zamíchají. Trénování se zpravidla zastavuje při konvergenci ztrátové funkce. Většina NN úkolů není konvexní, není tedy zajištěno optimální řešení [77].

Příklad jednoduchého klasifikačního modelu, který má dvě plně propojené vrstvy, aktivační funkci ReLU, softmax funkci a ztrátovou funkci křížové entropie je zobrazen na obrázku 3.4.



Obrázek 3.4: Jednoduchý model pro klasifikaci.

Problémy při trénování

Při trénování může dojít k problému, kde model úzce přilne k trénovací datové sadě a špatně zobecňuje pro jinou sadu. Tomuto problému se říká **overfitting**. Model sice dosahuje vysoké přesnosti při trénování, ale na validační sadě, kterou model při trénování přímo neviděl, už je přesnost nízká. Pro eliminaci overfittingu se používají regularizační techniky.

Dalším problémem je mizení nebo explodování gradientů. K mizení gradientů (anglicky **vanishing gradient**) dochází pokud jsou gradienty tak malé, že zabraňují aktualizaci hodnot vah. Mizení gradientů se řeší použitím určitých aktivačních funkcí (např. ReLU) a batch normalizace. Gradienty naopak mohou explodovat (anglicky **exploding gradient**). V tomto případě se gradienty nakumulují a výsledný velký gradient vyústí ve velké změny vah, což může vést k minutí minima (ideálního řešení). Řešením je přestavění modelu (např. použití LSTM architektury) nebo ořezávání gradientu (anglicky gradient clipping) [59].

U trénování s učitelem je potřeba datová sada, která reprezentuje reálnou distribuci pro daný úkol. Často jsou tedy vyžadovány velké anotované sady, které nemusí být jednoduché vytvořit. Trénovaní pomocí obrovských trénovacích sad je výpočetně náročné a často se používá i několik grafických procesorů [37]. Možným řešením je použití již před-trénovaných modelů. Díky technice *transfer learning* je možné modely již trénované pro nějaký úkol doladit a použít pro řešení jiného (ale podobného) úkolu [84].

Optimalizátory

Trénování neuronové sítě se provádí pomocí optimalizátoru, který hledá optimální řešení k definované ztrátové funkci. Cílem optimalizátoru je najít parametry modelu (váhy, biasy) s kterými se dosahuje minimální chyby na vzorcích trénovací sady.

Pro neuronové sítě se používají optimalizační metody, které pracují pouze s gradienty. Mezi takové optimalizační techniky patří Adaptive Moment Estimation (ADAM). ADAM počítá pro každý parametr adaptivní míru učení (anglicky learning rate) [36]. Pokud se gradient moc nemění, parametry se mění víc a naopak. Populární rozšíření jsou AdamW [42] a AMSGrad [72].

Regularizace trénování

Jednou z výhod neuronových sítí je jejich schopnost zobecňovat a tedy správně predikovat neznámá data [31]. K zajištění zobecňování se používá několik regularizačních metod, které jsou popsané v následující sekci. Problém, kdy model nezobecňuje dostatečně, se nazývá overfitting.

Váhový úbytek (anglicky weight decay) je regularizační technika, která zohledňuje váhy při výpočtu ztrátové funkce. Pokud označíme ztrátovou funkci $\mathcal{L}$, parametr váhového úbytku wd a součet čtverců všech vah L_2, pak se ztrátová funkce vypočítá jako 3.4. Váhový úbytek předchází overfittingu [38].

$$\mathcal{L} = \mathcal{L} + wd * L_2 \tag{3.4}$$

Další technikou pro regulaci je **dropout**. Cílem je během tréninku náhodně zahodit jednotky (neurony) z neuronové sítě a tím zabránit overfittingu. Často se zahazuje 20 - 50% jednotek [79].

Batch normalizace se zaměřuje na změny v distribuci parametrů modelu během trénování. Tento problém řeší pomocí normalizace vstupů vrstvy pro každou trénovací podmnožinu (batch). Kromě regulace také dovoluje vyšší rychlost učení [30].

Existující datovou sadu je možné rozšířit pomocí náhodného převracení, otáčení, změny měřítka, oříznutí, přeložení nebo přidáním šumu do daných záznamů. Díky tomu bude model robustnější. Tento přístup se nazývá **data augmentation** a používá se hlavně u obrazových dat [78].

3.3 Jazykové modely

Jazykový model (anglicky Language Model, LM) je sbírka statistických a pravděpodobnostních technik pro reprezentaci slov a pravděpodobností, se kterými se objevují dané sekvence slov. Jazykové modely se používají například pro strojový překlad [40] nebo rozpoznání řeči [50].

Používaní přímých indexů není pro reprezentaci slov efektivní, jelikož většina jazyků obsahuje tisíce slov. Proto se textová data reprezentují pomocí **word embeddingu**. Word embedding je vektorová reprezentace každého slova z předem definovaného slovníku. Slova jsou reprezentována v určité fixní dimenzi, která umožňuje zapouzdření vztahů mezi slovy.

Evaluace jazykových modelů

Pro hodnocení jazykových modelů se používá perplexita (anglicky perplexity, PPL). Perplexita je míra jak dobře pravděpodobnostní model predikuje určitý vzorek dat [32]. Čím

menší je perplexita, tím lepší je model. Perplexita záleží na textovém korpusu, jazykové modely musí být porovnávány na stejných datech.

Model q je vytvořený na základě trénovací datové sady. Model se může hodnotit mírou správných predikcí na testovací datové sadě $x_1, x_2, ..., x_N$. Pokud $\tilde{p}$ je empirické rozdělení testovacího vzorku, pak je křížová entropie definována jako 3.5.

$$H(\tilde{p}, q) = -\sum_{x} \tilde{p}(x) \log_2 q(x) \tag{3.5}$$

Perplexita se pak vypočítá pomocí křížové entropie jako 3.6. Základ logaritmu a exponenciální funkce nemusí být 2, ale musí být vždy stejný.

$$Perp(\tilde{p}, q) = 2^{H(\tilde{p},q)} \tag{3.6}$$

Lepší modely budou přiřazovat testovacím datům vyšší pravděpodobnosti $q(x_i)$ a to vyústí v malou perplexitu. Menší perplexita tady znamená, že model testovací vzorek 'víc zná'.

Statistické jazykové modely

Statistické jazykové modely (anglicky Statistical Language Models) přiřazují pravděpodobnosti pro sekvenci slov, kde ω_i je i-té slovo sekvence s a N je celkový počet slov v sekvenci s 3.7.

$$P(s) = P(\omega_1 \omega_2 ... \omega_N) \tag{3.7}$$

Pravděpodobnost je možné rozdělit na součin podmíněné pravděpodobnosti slov vzhledem k jejich předchůdcům 3.8. Těmto předchůdcům se říká kontext.

$$P(s) = P(\omega_1)P(\omega_2|\omega_1)...P(\omega_N|\omega_1\omega_2...\omega_{N-1}) \tag{3.8}$$

N-gram model

N-gram model je aproximační metoda, nepoužívá tedy celý kontext. $(k+1)$-gram model předpokládá, že slovo závisí pouze na k předchůdcích 3.9.

$$P(\omega_t|\omega_1...\omega_{t-1}) \approx P(\omega_t|\omega_{t-k}...\omega_{t-1}) \tag{3.9}$$

Nevýhodou n-gram modelů je, že přiřazují nulovou pravděpodobnost n-gramům, které se neobjevují v trénovací datové sadě. Existují techniky, které eliminují tento problém. Tyto techniky snižují pravděpodobnosti částem z trénovací sady a naopak přidávají pravděpodobnost částem, které se při trénování neobjevily.

Dalším zásadním problémem n-gram modelů je nedostatek dimenzionality, který omezuje modelování pro větších množství dat [33].

Neural Network Language Models

Neural Network Language Models (NNLM) jsou modely, které používají neuronové sítě k modelování jazyka. NNLM předčily statistické modely (pracující v diskrétním prostoru), protože díky neuronovým sítím je možné modelovat jazyk ve spojitém prostoru. To je možné díky distribuované reprezentaci slov [33].

Mezi základní modely pro reprezentaci slov NNLM patří CBOW a Skip-gram [48]. Tyto modely pracují s okolním kontextem pro predikování dalšího slova. Problémem modelů je limitovaná velikost kontextu, která se musí specifikovat před trénováním. Také nebere v úvah časový faktor slov v sekvenci.

RNN jazykové modely

Sekvenční data, jako jsou věty přirozeného jazyka, je možné zpracovávat pomocí rekurentních neuronových sítí (RNN), které se skládají z rekurentních vazeb [51]. Výstup sítě v daném časovém kroku slouží jako vstup modelu v následujícím časovém kroku. Díky tomu je možné vytvořit časově závislou neuronovou síť.

RNN jazykové modely (RNNLM) proto umožňují vstupy proměnné délky. Se změnou interního stavu se navíc zachová časová informace. Pro trénovací modely je ale náročné naučit se dlouhodobé závislosti. Je to kvůli tomu, že gradienty parametrů mohou během trénování mizet nebo explodovat, což vede ke zpomalení trénování nebo k nekonečné hodnotě parametrů.

LSTM-RNN jazykové modely

Kvůli problémům s mizícími a explodujícími gradienty se začala používat technologie Long-Short-Term-Memory (LSTM) [26]. LSTM kombinuje RNN blok s hradly. Tato hradla se během trénování učí, které informace si pamatovat, a které zapomenout.

LSTM-RNN jazykové modely tak pomocí hradel redukuje problém náročnosti učení se dlouhodobých závislostí.

Transformer

Transformer je neuronová architektura, která je založena na *attention* mechanismu (konkrétně se jedná o *multihead self attention mechanismus*) [83]. *Attention* mechanismy se aplikují ke shromažďování informací o relevantním kontextu daného slova. Tento kontext se zakóduje do vektoru reprezentujícího toto slovo.

RNNLM predikují další slovo pomocí jeho kontextu. Avšak ne každé slovo v kontextu souvisí s tímto slovem a není tedy efektivní pro jeho predikci. *Attention* mechanismus efektivně používá dlouhou historii vybíráním užitečných reprezentací slov [46]. Použita je sada *attention* koeficientů, pomocí kterých se získají cílové oblasti ze vstupu. Pomocí těchto koeficientů a reprezentací slov se vypočítá *attention* vektor, který se použije pro predikci dalšího slova.

Transformer zahrnuje enkodér a dekodér. Dekodér Transformeru je využit například v GPT (Generative Pre-trained Transformer) [68]. BERT, který je popsaný v následující sekci, používá zase enkodér Transformeru.

BERT

BERT (Bidirectional Encoder Representations from Transformers) je open source technika pro NLP vyvíjená společností Google [15]. BERT předčil předchozí metody, protože jako první používá plně obousměrné jazykové reprezentace.

Reprezentace mohou být buď **bezkontextové** (např. word2vec [49] nebo GloVe [62]) nebo **kontextové** (např. reprezentace LSTM-RNNLM nebo právě Transformer reprezentace). Kontextové reprezentace generují reprezentaci slova, která je na rozdíl od bezkontextové reprezentace založena na ostatních slovech ve větě.

Kontextové reprezentace se dále dělí na **jednosměrné** a **obousměrné**. O obousměrné reprezentaci se mluví, pokud je slovo reprezentováno jak vzhledem k předchozímu kontextu, tak i vzhledem k následujícímu kontextu. To znamená, že ve větě '*Vidím korunu stromu.*' bude slovo *korunu* reprezentováno jak ke slovu *vidím*, tak i ke slovu *stromu*. Jednosměrná reprezentace by slovo reprezentovala pouze k levému nebo pouze k pravému kontextu.

Předchozí práce (jedná se o práce založené na ELMo reprezentacích [63]) kombinovaly model používající levý kontext a model používající pravý kontext, ale zde se hovoří pouze o neúplné obousměrné reprezentaci. BERT reprezentace jsou společně trénovány na levém i pravém kontextu, proto se reprezentacím říká plně obousměrné [15].

Výsledkem je jazykový model, který je trénován na velkém textovém korpusu. Modely je možné rozdělit podle počtu vrstev a parametrů. Konkrétní model je následně možné doladit pro různé úkoly zpracování přirozeného jazyka jako je například analýza sentimentu, odpovídání na otázky nebo právě automatické hodnocení humoru. Samotný vyhledávač od Google používá BERT pro lepší porozumění uživatelským dotazům [56].

BERT byl trénován pouze pomocí korpusu obsahujícího prostý text. Konkrétně se jednalo o text z BookCorpus a z anglické Wikipedie. Je důležité, že se jedná o prostý neanotovaný text, jelikož se na webu nachází velké množství veřejně dostupného prostého textu ve více jazycích [15]. Kontext se v tomto případě získává pomocí Masked Language Model (MLM). Token se zamění za *[MASK]* a predikuje se pomocí všech předchozích i následujících tokenů [74].

3.4 Evaluační metody

K evaluaci systému pro zpracování přirozeného jazyka se často používají metriky F_1 (F-míra, anglicky F-measure) a přesnost (anglicky accuracy) [10]. Základem pro pochopení těchto metrik je **kontingenční tabulka záměn** 3.1. Pro každý vstup systém predikuje třídu (při tzv. *multiclass klasifikaci* může predikovat i více tříd). Třída může být například kategorie nebo stupeň vtipnosti. Skutečně pozitivní (**TP**) a skutečně negativní (**TN**) označují situace, kdy se systém zachoval korektně a třídu správně predikoval (třída měla být predikována) či správně nepredikoval (třída neměla být predikována). Naopak falešně pozitivní (**FP**) a falešně negativní (**FN**) říká, že třída neměla být predikována, ale byla nebo měla být predikována, ale nebyla (tedy opak správného stavu).

		Správný stav	
		Třída je predikována	*Třída není predikována*
Predikce systému	*Třída je predikována*	Skutečně pozitivní (anglicky true positive) **TP**	Falešně pozitivní (anglicky false positive) **FP**
	Třída není predikována	Falešně negativní (anglicky false negative) **FN**	Skutečně negativní (anglicky true negative) **TN**

Tabulka 3.1: Kontingenční matice záměn pro evaluaci NLP systému.

Přesnost

Přesnost (správnost, anglicky accuracy) měří kolik tříd bylo vybráno/nevybráno správně 3.10. Problém může nastat pokud jedna třída obsahuje dominantní počet vzorků. Tehdy může model pro každý vstup predikovat pouze jednu (dominantní) třídu a stejně dosahovat vysoké přesnosti.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{3.10}$$

$\mathbf{F_1}$

F_1 pracuje s **precizností** (anglicky precision) 3.11 a s **úplností** (senzitivita, anglicky recall) 3.12.

Preciznost určuje podíl skutečně pozitivních ze všech pozitivně predikovaných vzorků. Jedná se tedy o metriku, která určuje jak moc se dá věřit systému, že jeho predikce je správná.

$$Precision = \frac{TP}{TP + FP} \tag{3.11}$$

Úplnost počítá jaké je procento skutečně pozitivních ze všech vzorků, které pozitivní být měly. Jedná se tedy o metriku, která určuje jak moc se dá věřit systému, že predikoval všechny třídy, které měl.

$$Recall = \frac{TP}{TP + FN} \tag{3.12}$$

Díky preciznosti a úplnosti není problém s velkou mírou skutečně negativních (TN) tříd. Tyto metriky se totiž zaměřují výhradně na skutečně pozitivní (TP). Preciznost a úplnost jsou většinou nepřímo úměrné. Je tedy třeba nutné vhodně volit metriku pro evaluaci systému. Úplnost nebere v potaz falešně pozitivní (FP) třídy. Preciznost zase nepočítá s falešně negativními (FN) třídami.

F_1 3.13 poskytuje kompromis, protože počítá s přesností i úplností.

$$F_1 = \frac{2 * Precision * Recall}{Precision + Recall} \tag{3.13}$$

Kapitola 4

Návrh systému

Navržený systém by se měl skládat z webové aplikace, ve které je možné predikovat vtipnost a kategorii zadaného vstupního textu. Také by mělo být možné podat zpětnou vazbu k výsledkům systému. Cílem systému je schopnost ověřovat vtipnost a kategorii zadaných vtipů. Nejprve je popsán návrh klasifikátoru jak pro vtipnost, tak pro kategorii a následně výsledná webová aplikace.

4.1 Návrh klasifikátoru

Klasifikace vtipnosti i kategorie vychází z podobného návrhu, který je založen na neuronových sítích. Jedinými rozdíly jsou výstupní třídy a jejich počet. Tyto rozdíly jsou popsány v sekcích níže.

Při trénování se vstupní texty nejprve předzpracují tokenizátorem[1]. Výstupem tokenizátoru jsou **input ids** a **attention mask**. Input ids představují numerickou reprezentaci tokenů ze vstupu a attention mask říká, kterým tokenům by měl model věnovat pozornost.

Jako model pro klasifikaci je možné použít před-trénovaný BERT model (viz 3.3), ze kterého se extrahuje výstup pro klasifikační token.

Hlava klasifikátoru je navržena jako dvouvrstvá architektura. Dvě skryté vrstvy jsou spojeny nelineární funkcí ReLU. Výstupem jsou **logits**.

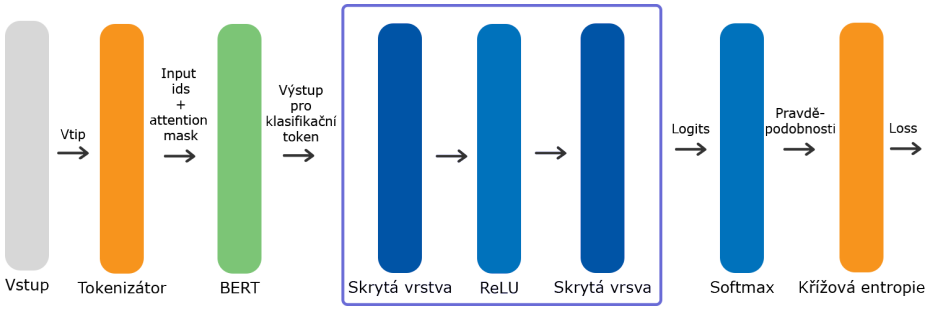
Logits jsou vstupem softmax funkce, která vrátí pravděpodobnost příslušnosti pro každou třídu. Nakonec je třeba vypočítat ztrátovou funkci pomocí křížové entropie, která určuje míru chyby mezi predikcí systému a očekávanou třídou. Této míře se říká **loss** a používá se pro ladění modelu. Výsledný návrh je zobrazen na obrázku 4.1. Jednotlivé části jsou popsány v sekci 3.2.

Pro natrénování už není třeba z logits počítat křížovou entropií loss, místo toho mohou vstoupit do *argmax* funkce, která vybírá jednu nejpravděpodobnější třídu. Tato třída představuje finální predikci systému.

Klasifikace vtipnosti

Většina současných prací klasifikuje humor pouze binárně na vtipné - nevtipné. Navržený klasifikátor rozpoznává tři třídy. První třída zahrnuje nevtipné vstupy a další dvě ty vtipné dle míry vtipnosti. Důvodem pro tří třídní klasifikaci je výhoda jemnějšího rozřazení vtipných vstupů, na které se systém zaměřuje.

[1] Tokenizace je proces segmentace textu. Text se rozdělí do jednotlivých částí, kterým se říká tokeny.



Obrázek 4.1: Návrh klasifikátoru pro automatické hodnocení humoru.

Zmíněné třídy jsou následující:

- *Not funny* - Nevtipné
- *Slightly funny* - Mírně vtipné
- *Very funny* - Hodně vtipné

Klasifikace kategorie

Kategorie představuje určitý centrální element, který je klíčovým konceptem na kterém vtip staví. Kategorie může být založena na stereotypech (například vtipy o blondýnkách) nebo na určitém typu vtipu (například krátký vtip, tzv. *one liner* nebo slovní hříčka - *pun*). I když se kategorie mohou teoreticky prolínat, v praxi se většinou kategorizují pouze do jedné kategorie.

Třídy pro kategorie vychází z nejčastějších kategorií vtipů vyskytujících se v použitých datových sadách. Celkově je systém schopný rozeznat 16 uvedených kategorií:

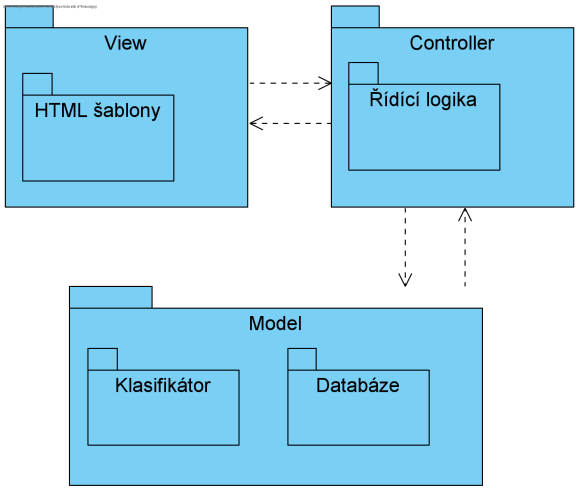
- Men / Women
- One Liners
- Animals
- Yo Mama
- Blonde
- Children
- Religious
- Puns
- Insults
- News / Politics
- At Work
- Medical
- Redneck
- Gross
- Bar
- Light Bulbs

4.2 Návrh webové aplikace

Cílem návrhu je vytvořit jednoduchou a uživatelsky přívětivou webovou aplikaci, která hodnotí uživatelský vstup podle vtipnosti a kategorie. Zároveň musí být systém schopen získat zpětnou vazbu uživatele pro případné další doladění klasifikátoru.

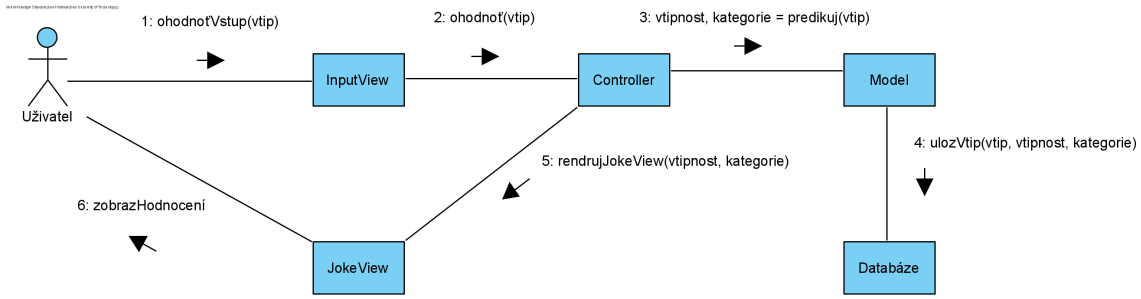
Pro systém je zvolen architektonický vzor Model View Controller (MVC) 4.2, který se hojně používá právě pro webové aplikace. MVC si zakládá na oddělní logiky od výstupu. **Model** obsahuje logiku, tedy převážně výpočty a databázové dotazy. **View** má na starost zobrazení výstupu uživateli, obsahuje html šablonu a minimální množství logiky. **Controller** představuje prostředníka mezi uživatelem, modelem a view [22].

Uživatel tedy komunikuje s view. View převezme od uživatele vstup a předá ho controlleru, který pomocí modelu získá jeho ohodnocení. Pro nové vstupy se hodnoty získají z klasifikátoru popsaného výše. Jelikož odhadování vtipnosti a kategorie může být časově



Obrázek 4.2: Návrh architektury systému pomocí Model View Controller (MVC).

náročné, již ohodnocené vstupy se uloží i s predikcemi do databáze, ze které je následně možné již predikované vstupy rychle získat. Poté, co controller získá od modelu predikce, vyrendruje view, které se předá zpět uživateli. Predikce kategorie se nevykresluje pokud je vstup klasifikován jako nevtipný. Diagram komunikace pro tento scénář je zobrazen na obrázku 4.3.



Obrázek 4.3: Komunikační diagram pro hodnocení vstupu od uživatele.

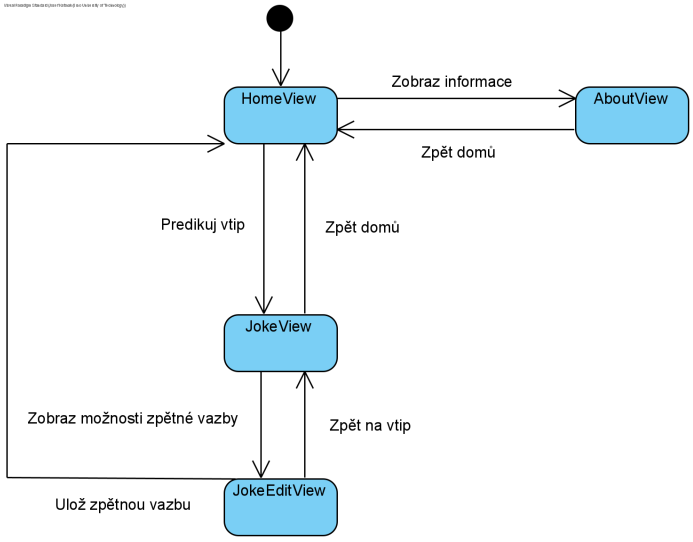
Podobně systém bude fungovat při poskytnutí uživatelské zpětné vazby. Z view controller obdrží zpětnou vazbu, kterou uloží do databáze v modelu a vrátí uživateli příslušné view s informací o uložení zpětné vazby.

Kromě domovského view, kam uživatel zadává vstup, view s predikcí vtipu a view pro zpětnou vazbu obsahuje návrh ještě další view s informacemi o projektu. Jak na sebe view navazují je zobrazeno na diagramu návaznosti obrazovek 4.4.

Databáze

K systému je navrhnuta databáze pro uchování již predikovaných vstupů a zpětné vazby od uživatele. Databáze obsahuje dvě zásadní entity: vtip a kategorii. U kategorie se uchovává pouze jméno kategorie a tato entita se může předem naplnit daty (kategorie jsou předem určené).

Entita pro vtip zahrnuje tělo vtipu, informaci, zda se vtip již predikoval a případně predikovanou vtipnost a kategorii. Predikovaná vtipnost se ukládá jako hodnota 0 až 2

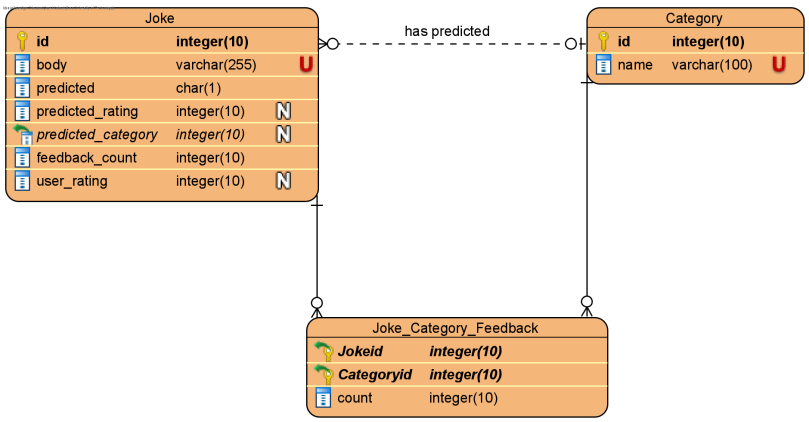


Obrázek 4.4: Diagram návaznosti obrazovek webové aplikace.

(jsou tři třídy vtipnosti), predikovaná kategorie obsahuje id entity pro danou predikovanou kategorii. Pro zpětnou vazbu entita vtipu vlastní atributy pro součet hodnocení vtipnosti ze zpětných vazeb a celkový počet zpětných vazeb. Z těchto údajů je pak možné vypočítat průměrné hodnocení vtipnosti zpětných vazeb.

Zpětná vazba pro kategorie vtipu se řeší N:M vazební entitou, která ještě navíc pro kombinaci vtip-kategorie uchovává kolikrát byla tato kombinace zpětnou vazbou poskytnuta.

Entity-relationship diagram (ERD) databáze je zobrazen na obrázku 4.5.



Obrázek 4.5: ERD pro databázi systému.

Kapitola 5

Implementace

Pro implementaci systému je zvolen vysokoúrovňový skriptovací programovací jazyk Python. Stejně jako u návrhu systému je v této kapitole nejprve popsán klasifikátor a poté samotná webová aplikace.

5.1 Implementace klasifikátoru

Příprava datové sady

Pro trénování klasifikátoru je nejprve třeba připravit data z datové sady. Data jsou načtena pomocí datové struktury DataFrame z knihovny *pandas* [44]. DataFrame se pošle do BERT tokenizátoru, který je zpřístupněn pomocí knihovny *transformers* [87]. Z tokenizátoru se získají input ids a attention masks, na které se použije *PyTorch* DataLoader [60]. DataLoader se stará o iteraci nad datovou sadou a podporuje vzorkování, automatické zamíchání a rozdělování dat do batches.

Maximální délka tokenů pro vstup, velikost batche a rozhodnutí zda vstup převádět na malé písmena se nastavuje pomocí konfiguračního souboru popsaného níže. Pokud je vstup delší než maximální délka, tak se jeho pravá strana ořízne a zahodí. Pokud je vstup kratší než maximální délka, tak se pomocí speciálních tokenů na maximální délku zarovná[1], protože model vyžaduje fixní délku vstupu.

Klasifikátor

Dvouvrstvá architektura hlavy klasifikátoru je uzavřena v sekvenčním kontejneru, který obsahuje dvě lineární skryté vrstvy mezi kterými se nachází nelineární ReLU funkce. Velikost vstupu první skryté vrstvy se rovná velikosti výstupu použitého modelu pro klasifikační token a výstup druhé skryté vrstvy má velikost rovnu počtu klasifikovatelných tříd. Velikost výstupu první a vstupu druhé skryté vrstvy je určena parametrem v konfiguračním souboru (viz níže). Architektura je založena na knihovně *PyTorch* [60]. Počet tříd se liší podle toho, zda se klasifikuje vtipnost nebo kategorie, a je ho možné nastavit také v konfiguračním souboru.

Při forward-pass se tokenizované vstupy pošlou do modelu. První token každého vstupu je vždy speciální klasifikační token *[CLS]*. Z *[CLS]* tokenu se extrahuje poslední skrytý stav představující souhrnou reprezentaci vstupu pro účely klasifikace. Tento stav se nakonec vloží do klasifikátoru pro výpočet logits. Potřebné kroky pro výpočet logits jsou popsány

[1] *Attention mask* určuje, kterým tokenům se má věnovat pozornost a které byly přidány pro zarovnání.

algoritmem 1. Jako model je využit BERT model, který je již před-trénovaný a je dostupný z knihovny *transformers* [87]. Knihovna obsahuje několik před-trénovaných modelů podle počtu vrstev a parametrů. V konfiguračním souboru systému je možné zvolit, který z nich se pro klasifikaci použije.

Algorithm 1 Výpočet logits

```
model ← loadBertModel()
modelOutSize ← getOutputSize(model)
hiddenSize ← getParam('CLASSIFIER_HID_SIZE')
classesSize ← getParam('NUMBER_OF_CLASSES')
firstHiddenLayer ← Linear(inputSize = modelOutSize, outputSize = hiddenSize)
secondHiddenLayer ← Linear(inputSize = hiddenSize, outputSize = classesSize)
classifierHead ← Sequential(firstHiddenLayer, ReLU(), secondHiddenLayer)
CLSHiddenState ← model(inputIds, attentionMask)
logits ← classifierHead(CLSHiddenState)
```

Inicializace pro trénování

Před startem trénování se nastaví seed pro možnost reprodukce výsledků a inicializuje se klasifikátor, optimalizátor, plánovač (anglicky scheduler) a ztrátová funkce. Seed, počet epoch a parametry optimalizátoru je možné změnit v konfiguračním souboru.

Klasifikátor je reprezentován jako třída s funkcí pro forward-pass. Při inicializaci je možné načíst stav již laděného modelu.

Optimalizátor a plánovač jsou zpřístupněny knihovnou *transformers* [87]. Jako optimalizátor je použitý AdamW, který přijímá dva konfigurovatelné parametry - míru učení a epsilon. Plánovač dovoluje ladění míry učení v průběhu trénování. Systém používá lineární plánovač, který lineárně zmenšuje míru učení. Cílem jsou velké změny vah na začátku trénování, ale ke konci trénovacího procesu už jsou změny menší pro snadnější doladění modelu.

CrossEntropyLoss funkce z *PyTorch* knihovny [60] je použita pro ztrátovou funkci. Tato funkce v sobě zároveň obsahuje i softmax funkci.

Trénovací smyčka

Trénování podle počtu epoch několikrát iteruje přes celou trénovací datovou sadu. Pro každou batch se načtou data, vynulují gradienty a vypočítají logits. Z logits a očekávaných tříd se vypočítá pomocí ztrátové funkce loss. Z loss se pak získají gradienty. Gradienty se ořízno kvůli prevenci explodujících gradientů. Nakonec se provedou kroky optimalizátoru a plánovače, které aktualizují parametry a míru učení.

Podle parametru z konfiguračního souboru se po určitém počtu batches provede evaluace. V evaluaci se používá validační datová sada, ze které se vypočítá validační pruměrná loss, přesnost a F_1. Funkce pro výpočet přesnosti a F_1 jsou dostupné v knihovně *scikit-learn* [61] a systémem predikované třídy se pro tyto metriky získají funkcí *argmax*. Pokud model dosáhl lepší loss než je dosavadní nejlepší, tak se model uloží. Ukládání modelu je možné předem definovat i na základě nejlepší přesnosti nebo F_1.

Po celou dobu trénování se postupně vypisují informace pro epochu, batch, trénovácí loss, validační loss, validační přesnost, validační F_1 a uběhlý čas. Na konci trénování se navíc vypíše nejlepší validační ztráta, přesnost a F_1.

Jádro trénovací smyčky je popsáno algoritmem 2.

Algorithm 2 Trénovací smyčka

```
model ← loadModel()
bestLoss ← int.maxsize
for all epoch ∈ epochs do
    step ← 0
    for all batch ∈ batches do
        model.zeroOutGradients()
        logits ← model.forwardPass(batch.inputIds, batch.attentionMasks)
        loss ← crossEntropyLoss(logits, batch.trueClasses)
        gradients ← loss.backward()
        gradients ← clipGradients(gradients)
        model.params ← optimizer.step(gradients)
        optimizer.learningRate ← scheduler.step(optimizer.learningRate)
        if step mod EVALUATE_AFTER then
            validationLoss ← evaluate(model)
            if validationLoss < bestLoss then
                save(model)
                bestLoss ← validationLoss
            end if
        end if
        step ← step + 1
    end for
end for
```

Evaluace a predikce

Systém umožňuje hodnocení modelu na testovací datové sadě, která se vůbec nepoužívá při trénování. Evaluace probíhá podobně jako trénování, ale provádí se pouze forward-pass a logits se získají bez výpočtu gradientů. Nakonec se nepočítá ztrátová funkce, ale rovnou metriky přesnost a F_1.

Výsledné predikování použité ve webové aplikaci bere pouze jeden vtip jako vstup. Vstup se předá prvnímu modelu trénovanému pro predikci vtipnosti a následně druhému modelu trénovanému pro predikci kategorie. Výsledkem jsou dvě predikce - pro vtipnost a pro kategorii. Modely jsou pro zrychlení výpočtu načteny pouze jednou při startu webové aplikace.

Konfigurovatelné parametry

Pro klasifikaci se v rámci celého systému berou informace z konfiguračního JSON souboru *hyperparams.json*, který obsahuje následující parametry:

- *BERT_MODEL* - použitý BERT model
- *SEED* - celočíselná hodnota pro reprodukovatelnost výsledků
- *MAX_LEN* - maximální počet tokenů pro vstup

- *LOWER_CASE* - zda převádět/nepřevádět vstup na malá písmena
- *BATCH_SIZE* - počet záznamů použitých v jedné iteraci trénování
- *EPOCHS* - počet iterací přes celou datovou sadu
- *LEARNING_RATE* - míra učení pro optimalizátor
- *EPSILON* - malá hodnota pro zabránění dělení nulou u optimalizátoru
- *CLASSIFIER_HID_SIZE* - hodnota pro velikost skrytých vrstev
- *NUMBER_OF_CLASSES* - počet predikovaných tříd
- *EVALUATE_AFTER* - po kolika batches se provádí evaluace

Tyto parametry se ladí pro nejlepší výsledek tréninku. Ladění klasifikátoru probíhá pomocí experimentů, kterým se věnuje kapitola 7.

5.2 Implementace webové aplikace

Pro vývoj webových aplikací v jazyku Python se často používá framework *Django* nebo knihovna *Flask*. Pro implementaci systému byl zvolen Django framework [16], protože je oproti Flask knihovně pokročilejší.

Struktura implementované aplikace

Django využívá architekturu *MVC* (viz 4.2), ale pojmenovává jednotlivé části architektury jinak. Upravená verze architektury se nazývá *MVT* (Model View Template). Model stejně jako v původní verzi obsahuje logiku. Template představuje View část z MVC a obsahuje šablony, které slouží k zobrazení výstupu. Controller z MVC se v MVT jmenuje View a představuje prostředníka mezi Modelem a Templates.

Django se skládá z tzv. aplikací, které fungují samostatně a jsou znovupoužitelné. Jelikož systém vyžaduje pouze jednoduchou webovou aplikaci, skládá se z pouze jedné aplikace, která je nazvána *ahh*. V aplikaci se se nachází modely, views i templates.

Model

Django pracuje s *ORM* (Object Relational Mapping), díky kterému je možné přistupovat k relační databázi skrze Django objekty. Django také poskytuje pro přístup k databázi uživatelské rozhraní a zabezpečuje systém před SQL injection útoky. Použita je *SQLite* relační databáze, protože je vhodná pro malé projekty. Soubor *models.py* obsahuje definici tabulek databáze podle návrhu z 4.2.

Veškerý kód klasifikátoru je uložen v adresáři *src*. Adresář obsahuje balíček *predict* se skriptem *predict.py*, který se používá pro získání konečné predikce vtipnosti i kategorie vstupu.

Templates

Výstup je pro uživatele vykreslován z *HTML* (HyperText Markup Language) šablon, které se před předáním uživateli naplní daty. Šablony se nachází v adresáři *templates*. Aplikace obsahuje následující HTML šablony:

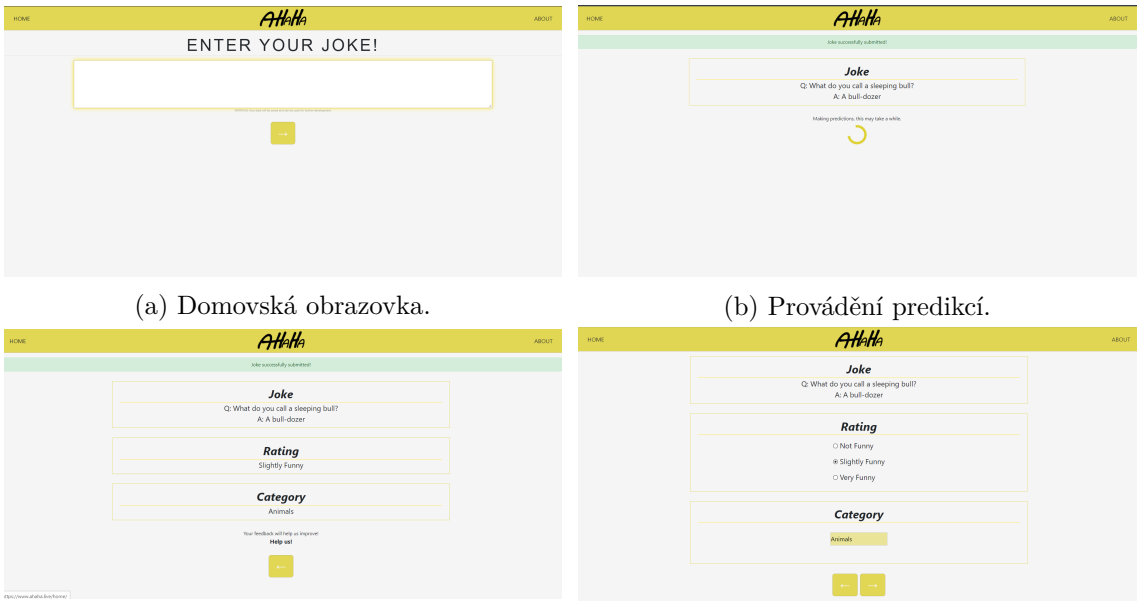
- 404.html
- 500.html
- about.html
- base.html
- home.html
- joke.html
- joke_edit.html

V *base.html* se nachází základní struktura stránky, která je použita pro všechny ostatní šablony. Základní struktura obsahuje navigační panel a zabezpečuje načtení všech částí stránky jako například CSS stylů nebo JavaScript skriptů. Ostatní šablony rozšiřují *base.html* o další obsah. Šablona *home.html* dovoluje uživateli vložit vstup, *about.html* vypisuje základní informace o projektu, *joke.html* vypisuje predikovanou vtipnost a kategorii a k získání zpětné vazby od uživatele slouží *joke_edit.html*. Chybové šablony *404.html* a *500.html* informují o daných chybách.

Vzhled stránek je zajištěn *CSS* (Cascading Style Sheets) styly. Kromě vlastních CSS stylů aplikace využívá i knihovnu *Bootstrap*, která dovoluje využití již předem vytvořených CSS stylů.

Jelikož predikování tříd pro vstup může trvat delší dobu, systém obsahuje *JavaScript* kód s technologií *AJAX*, která umožňuje interaktivní zobrazení *spinneru* s informací, že klasifikace může zabrat nějaký čas. Po získání predikcí se na stránce výsledek automaticky zobrazí a informace o provádění výpočtů zmizí.

Vlastní CSS styly, JavaScript soubor a další neměnné soubory použité k renderování stránek (favicon, obrázky) jsou obsaženy ve složce *static*. Výsledné uživatelské rozhraní je zobrazeno na obrázku 5.1.



(a) Domovská obrazovka.

(b) Provádění predikcí.

(c) Vypsání predikcí.

(d) Poskytování zpětné vazby.

Obrázek 5.1: Uživatelské rozhraní systému.

Views

Soubor *views.py* propojuje modely a šablony. Při zadání URL adresy Django provádí routování, které je nastaveno v souboru *urls.py*. Routování zajišťuje volání správné view funkce.

Systém přebírá vstup od uživatele pomocí formulářů. Celkově obsahuje tři formuláře: formulář pro vstupní vtip a dva formuláře pro zpětnou vazbu. Zpětná vazba se získává jak pro vtipnost, tak pro kategorii. Tyto dva formuláře jsou předem naplněny možnostmi, které představují jednotlivé klasifikační třídy. Uživatel se může rozhodnout podat zpětnou vazbu a vybrat podle svého uvážení z předem připravených možností vtipnost a kategorii pro zadaný vstup.

Pro každou HTML šablonu existuje ve *views.py* funkce, která šablonu vykresluje a případně zpracovává vstup z formulářů. Každá funkce dostává na vstupu parametr *request*, který obsahuje metadata o požadavku na server.

Téměř všechny funkce komunikují s modelem. Pokud uživatel vloží vstup pro predikování, nejprve view z modelu dotazem do databáze zjistí, jestli už vstup nebyl predikován. Pokud byl, model predikce view předá a view vyrendruje příslušnou šablonu.

V případě, že se vstup ještě nepredikoval, se zavolá klasifikátor a vykreslí se stejná šablona s informací o provádění predikcí. Jakmile klasifikátor predikce vrátí, tak se tyto predikce uloží do databáze pro další použití a zobrazí se na již vykreslené šabloně, což je možné díky AJAX technologii, která komunikuje se speciální view funkcí, jejíž výsledek dokáže dynamicky zobrazit.

Pokud se uživatel rozhodne poskytnout zpětnou vazbu, view informuje model, který ji uloží. Následně toto view vykreslí domovskou stránku, kde uživatel může vložit další vstup. Funkce pro vypsání a získání predikcí a zpětné vazby přijímají kromě *request* parametru také *joke_id* parametr, který jim usnadňuje komunikaci s modelem.

Nasazení webové aplikace

Systém je nasazen na linuxový server, který běží na virtuálním stroji zprostředkovaném díky Microsoft Azure. Aplikace je nasazena pomocí *Apache* a *ModWSGI*. Apache je softwarový webový server a ModWSGI je modul, který umožňuje nasazení Python aplikací.

Web server podporuje HTTPS díky SSL/TLS certifikátu a má přiřazené doménové jméno. URL adresa pro přístup k webové aplikaci je `https://www.ahaha.live`.

5.3 Správa kódu a testy

Pro veškerou správu kódu byl použit distribuovaný systém správy verzí *Git*, konkrétně webový Git repozitář *GitLab*. GitLab zároveň podporuje sledování chyb a nedokončených částí, čehož bylo využito při vývoji systému.

Systém obsahuje automatické testy jak pro klasifikátor, tak pro webovou aplikaci. Klasifikátor je možné testovat díky frameworku *unittest*. Pro webovou aplikaci má django vlastní testovací framework založený na *unittest*.

Pro generování dokumentace obsahuje kód *docstrings*, které dokumentují funkce a třídy systému. Automatickou dokumentaci je možné vygenerovat pomocí Sphinx.

Kapitola 6

Datové sady

Datové sady jsou důležité pro trénování modelu určeného k automatickému hodnocení humoru. Existuje několik různých datových sad, které souvisí s problematikou hodnocení nebo rozpoznání humoru v textu. Vytváření korpusu pro hodnocení humoru není jednoduché, jelikož zahrnuje lidskou anotaci a jasnou definici humoru k dosažení dohody mezi anotátory. V následující sekci uvedu a popíšu některé z existujících datových sad.

6.1 Existující datové sady

Humicroedit dataset

Následující datová sada byla poskytnuta k úloze posuzování humoru v lehce editovaných anglických novinových titulcích (SemEval-2020 Task 7) [52]. V novinových titulcích lidé vybraní pomocí crowd-sourcingu změnili slovo tak, aby byl titulek vtipný. Editoři byli pověřeni, aby titulky upravili způsobem, který přijde vtipný široké veřejnosti. Následná vtipnost byla určena také pomocí crowd-sourcingu. Novinové titulky byly získány z Redditu a byly upraveny a hodnoceny lidmi z Amazon Mechanical Turk[1]. Datová sada celkově obsahuje 15,095 upravených titulků.

Mezi nejčastější strategie při úpravě titulku patří:

1. Použití slova, které tvoří smysluplný n-gram s okolními slovy.
2. Použití slova, které je sémanticky vzdálené nahrazovanému slovu.
3. Použití slova, které má silnou vazbu na entitu v titulku.
4. Použití sarkasmu.
5. Znevažování entity nebo slova (podstatného jména) v titulku.
6. Potlačení napětí, zesměšňování vážných titulků
7. Použití slova vyvolávajícího nesoulad.
8. Použití pointy: titulek směřuje k očekávanému konci, kde se nachází změněné slovo, které vytváří souvislý, ale překvapivý konec.

[1] Amazon Mechanical Turk je webová stránka pro crowd-sourcing. Žadatelé si zde mohou vzdáleně najmout pracovníky pro své vlastní úlohy.

Datová sada pomáhá ke zkoumání, jak malé změny aplikované na text mohou tento text změnit z nevtipného na vtipný. Díky tomu je možné se zaměřit na humorný efekt atomických změn a zlomový bod mezi normálním a humorným textem.

Na rozdíl od některých jiných datových sad, které se zaměřují pouze na binární klasifikaci vtipné-nevtipné, novinové titulky této sady byly hodnoceny známkou 0-3, kde vyšší známka značí větší vtipnost titulku. Titulky hodnotilo pět kvalifikovaných porotců. Datová sada obsahuje známky a celkové hodnocení, které je průměrem hodnocení od porotců. Příklady z datové sady jsou zobrazeny v tabulce 6.1 [27].

K úloze byla poskytnuta také datová sada získaná z interaktivní hry **FunLines** [28]. Hráči účastnící se hry měnili novinové titulky a hodnotili vtipnost upravených titulků ostatních hráčů. Datová sada má stejnou strukturu jako Humicroedit a obsahuje 8,248 upravených titulků.

Původní titulek	**Náhrada**	**Známka**
Kushner to visit **Mexico** following latest Trump tirades	therapist	2.8
4 arrested in Sydney raids to stop terrorist **attack**	kangaroo	2.6
The Latest: BBC cuts **ties** with Myanmar TV station	pies	1.8
Congress Achieves the Impossible on **Tax** Reform	toilet	0.8
4 **soldiers** killed in Nagorno-Karabakh fighting: Officials	rabbits	0.0

Tabulka 6.1: Příklady z datové sady Humicroedit. Tučné slovo z původního titulku je nahrazeno náhradou. Známka je průměr hodnocení pěti porotců.

Hodnocení vtipnosti slov

Datová sada obsahuje 4,997 anglických slov s ohodnocením vtipnosti každého slova. Hodnocení byla provedena pomocí crowd-sourcingu. Slova byla hodnocena známkou 1-5, kde vyšší známka znamená vyšší vtipnost slova [17]. Datová sada obsahuje individuální hodnocení každého hodnotícího a navíc informace o konkrétním hodnotícím. Mezi tyto informace patří pohlaví, věk a vzdělání. Hodnoticí byli vybrání z Amazon Mechanical Turk. Každé slovo bylo hodnoceno minimálně patnácti hodnotícími a průměrně třiatřiceti hodnotícími.

Ironické/sarkastické recenze

Datová sada obsahuje klasické a ironické/sarkastické recenze produktů z Amazonu. Pro vytvoření datové sady byl využit crowd-sourcing (Amazon Mechanical Turk) [20]. V datové sadě se nachází také páry recenzí pro stejný produkt, kde jedna není a druhá je ironická/sarkastická. Detekce sarkasmu je možná na úrovni dokumentu a na úrovni vět. Datová sada se zaměřuje na detekci sarkasmu a ironie v celém dokumentu. Na rozdíl od detekce sarkasmu nebo ironie na úrovni vět je v dokumentu přítomen širší kontext. Sada obsahuje 1,905 anotovaných dokumentů s recenzemi.

SARC (Self-Annotated Reddit Corpus)

SARC je velký anotovaný korpus pro detekci sarkasmu. Korpus obsahuje 1.3 milionů sarkastických záznamů. Navíc obsahuje několikanásobně víc ne-sarkastických záznamů. Sarkasmus je určen autorem záznamu. Navíc korpus obsahuje uživatele, téma a konverzační kontext záznamu [35]. Záznamy pochází z webové stránky Reddit.

Humor Detection: A Transformer Gets the Last Laugh

Datová sada obsahuje vtipy ze tří zdrojů - vtipy dlouhé pár vět, slovní hříčky o délce jedné věty a různorodé vtipy z Redditu. Data z Redditu jsou rozdělené na tělo vtipu a pointu. Podle hodnocení uživatelů jsou vtipy rozděleny na vtipné a nevtipné. Část s krátkými vtipy obsahuje datovou sadu s vtipy z Kaggle a nevtipné novinové titulky s podobným rozloženým slov a znaků. Poslední část datové sady obsahuje 16,001 slovních hříček a 16,002 vět, které slovní hříčku neobsahují, ty jsou získány také z novinových titulků [85].

Making Computers Laugh: Investigations in Automatic Humor Recognition

Datová sada se zaměřuje na jednořádkové vtipy. Vtip má obvykle délku patnácti či méně slov. Vtipy jsou získány pomocí webového bootstrapingového algoritmu. Výsledná sada obsahuje circa 16,000 vtipů, které jsou doplněny strukturálně podobnými, ale nevtipnými záznamy. Mezi zdroje pro nevtipné záznamy patří Reuters, Proverbs a British National Corpus (BNC) [47].

#HashtagWars

Datová sada obsahuje tweety, které jsou vtipnou odpovědí pro hashtag zadaný v rámci televizní show @midnight. Tato show poté vyhlašuje deset nejvtipnějších tweetů. Celkově bylo sesbíráno 9,658 tweetů pro 86 hashtagů. Tweety poté spadají do tří kategorií: vítězný (a tedy nejvtipnější) tweet, tweety, které se umístily v nejlepší desítce nejvtipnějších tweetů (ale nevyhrály) a zbytek tweetů. Datová sada byla poskytnuta pro SemEval-2017 Task 6 [66].

Anglické slovní hříčky

Jedná se o dvě datové sady poskytnuté k úloze detekce a interpretace slovních hříček (SemEval-2017 Task 7) [53].

První datová sada obsahuje homografické záznamy. Konkrétně se jedná o slovní hříčky, vtipy, aforismy a jiné krátké samostatné záznamy od profesionálních humoristů a z online sbírek. Slovní hříčky musí mít oproti určité lexikální jednotce jiný význam, ale musí se hláskovat zcela stejným způsobem (můžou se ignorovat částice a skloňování). Tato datová sada obsahuje 2,250 záznamů, z čehož 1,608 obsahuje slovní hříčku. Anotaci provedli tři vyškolení lidé.

Druhá datová sada je podobná té první, ale jedná se spíše o heterografické slovní hříčky. Heterografické slovní hříčky spoléhají na podobné, na rozdíl od homografických ne přímo identické, hláskování slov. Heterografické slovní hříčky jsou tedy více komplexní. Tato datová sada obsahuje 1,780 záznamů, z čehož 1,271 obsahuje slovní hříčku.

6.2 Datové sady použité v systému

Základem pro vytvoření datové sady k systému je *A dataset of English plaintext jokes* [67]. Datová sada obsahuje okolo 208,000 anglických textových vtipů. Vtipy pocházejí ze tří zdrojů. Pro každý zdroj má datová jednotka jinou strukturu. První zdrojem je *Reddit*, kde sada obsahuje tělo a skóre vtipu. Skóre zde představuje počet hlasů od uživatelů Redditu. Tento zdroj je nejobsáhlejší a obsahuje cca 195,000 záznamů. Nevýhodou je, že neobsauje

kategorii ani normalizované hodnocení. Druhým zdrojem je webová stránka *stupidstuff.org*. Zde sada kromě samotného vtipu obsahuje i kategorii a hodnocení 0-5, tedy všechny důležité informace potřebné pro systém. Bohužel je ale s cca 3,700 vtipy nejmenší. Poslední zdroj o velikosti cca 10,000 záznamů z *wocka.com* obsahuje vtip a jeho kategorii.

Datová sada je získána pomocí web scrapingu a kromě samotných JSON souborů pro každý zdroj obsahuje také Python skript použitý pro scraping.

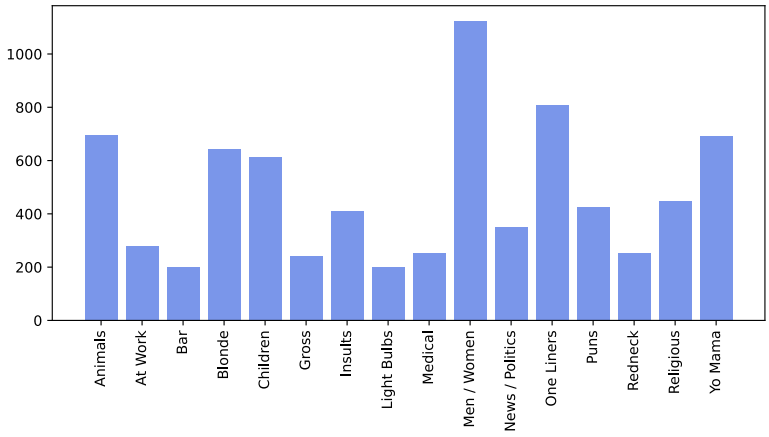
Datová sada pro rozpoznání kategorie

Pro rozpoznání kategorie vtipu jsou použity dva zdroje ze zmíněné datové sady. První datová sada obsahuje vtipy ze stránky *stupidstuff.org* a druhá ze stránky *wocka.com*. Skript pro web scraping je upraven, aby pro *wocka.com* získával i hodnocení vtipu. JSON soubory pro každý zdroj obsahují tělo, kategorii a hodnocení vtipu.

Wocka datová sada je kvůli šumu vyfiltrována a použity jsou pouze záznamy s hodnocením vyšším než 0,5. Pro účely klasifikace vtipnosti mohou být tyto záznamy užitečné, protože obvykle neobsahují vtip a mají velice nízké hodnocení. Ale pro klasifikaci kategorie se nehodí, jelikož tělo vtipu neodpovídá jeho kategorii.

Datové sady jsou načteny a spojeny do jedné. Z vytvořené datové sady jsou odstraněny záznamy s prázdným tělem vtipu. Následně jsou upraveny názvy kategorií vtipů tak, aby vtipy s dosti podobným názvem kategorie spadaly do stejné kategorie. Kategorie pro ostatní/smíšené vtipy a kategorie, které mají méně než 200 záznamů jsou odstraněny. Poté se vtipy seskupí dle kategorie a náhodně se rozdělí na trénovací, validační a testovací část. Trénovací část obsahuje circa 80% záznamů, testovací a validační část každá circa 10% záznamů.

Trénovací datová sada obsahuje 6,110 záznamů, testovací a validační datová sady obsahují obě 764 záznamů. Nakonec se odstraní přebytečné atributy, aby každý záznam obsahoval pouze kategorii a tělo vtipu. Datová sada obsahuje 16 rozpoznávaných kategorií. Výsledné kategorie vychází z kategorií, které se používají na stránkách s vtipy ze kterých pochází datová sada. Konkrétně se jedná o následující kategorie: Animals, At Work, Bar, Blonde, Children, Gross, Insults, Light Bulbs, Medical, Men / Women, News / Politics, One Liners, Puns, Redneck, Religious, Yo Mama. Počty vtipů pro jednotlivé kategorie jsou zobrazeny na histogramu 6.1.



Obrázek 6.1: Histogram počtu vtipů pro jednotlivé kategorie.

Datová sada pro hodnocení vtipnosti

Pro hodnocení vtipnosti je kromě výše zmíněné datové sady s textovými vtipy použita i sada *News Commentary* [82], která obsahuje komentáře z novinových článků.

Nejprve se vezmou části z *wocka.com* a *stupidstuff.org*. Datové sady mají rozdílné hodnocení vtipnosti, *stupidstuff.org* používá rozsah 0-5, ale *wocka.com* hodnotí na škále 0-3. Hodnocení se tedy zvlášť pro každý zdroj normalizuje do tří tříd - nevtipné, mírně vtipné a hodně vtipné.

Následně se řeší nevyváženost tříd. Mírně vtipných vtipů je nejvíc a to circa 7,000. Třída nevtipných záznamů se doplní právě daty z *News Commentary*. Pro zachování podobné struktury se jako záznam náhodně vyberou 1 až 3 řádky z této datové sady. Velmi vtipná třída se doplní z *Reddit* datové sady a to záznamy s nejvyšším skóre. Díky tomu se datová sada vyrovná tak, aby každá třída obsahovala cca 7,000 záznamů.

Stejně jako u datové sady pro klasifikaci kategorie se nakonec odstraní záznamy s prázdným tělem. Výsledkem je 20,775 záznamů, které se nakonec náhodně rozdělí do trénovací (80%), validační (10%) a testovací (10%) sady.

Kapitola 7

Experimenty

Experimenty souvisí s trénováním a evaluací modelů pro klasifikaci vtipnosti do tří tříd a kategorie do šestnácti tříd. Návrh klasifikátoru je popsán v sekci 5.1 a jeho implementace v sekci 4.1. Trénování i testování probíhá pomocí vytvořených datových sad uvedených v sekci 6.2.

Jednodušší trénování probíhalo lokálně na grafické kartě NVIDIA GeForce GTX 1050 Ti. Složitější výpočty už byly přesunuty na výpočet v cloudu a to v prostředí *Google Colaboratory*. *Google Colaboratory* nabízí hardwarovou akceleraci pomocí GPU. Model GPU je automaticky přidělen, není tedy možné si ho explicitně vybrat. Konečné predikce se provádí na virtuálním stroji, který obsahuje dvoujádrové CPU. Modely se načítají pouze jednou při startu webové aplikace. Rychlost výpočtu konečných predikcí jednoho vzorku pro uživatele zpravidla trvá maximálně dvě vteřiny.

7.1 Baseline

Pro budoucí porovnání s výsledky klasifikátoru byla nejprve vytvořena baseline. Baseline používá *TF-IDF vectorizer* a *Naive Bayes Classifier*. Nejprve se použije *TF-IDF* k vektorizaci vstupních dat a poté *Naive Bayes* model jako klasifikátor. *Naive Bayes Classifier* je jednoduchý, ale efektivní lineární klasifikátor, který je použitelný pro textová data [70]. Vectorizer i klasifikátor jsou zpřístupněny knihovnou *sklearn* [61].

Vstupní data se předpracují pro *bag-of-words* model, který nebere v potaz gramatiku a pořadí slov. *TF-IDF* (Term Frequency–Inverse Document Frequency) vektorizuje data do reprezentace vhodné pro klasifikátor, která odráží významnost jednotlivých slov. *Naive Bayes Classifier* má pouze jeden hyper-parametr nazvaný **alpha**. Tento parametr se pomocí trénovacích dat ladí k získání nejlepší přesnosti. Nejlepší alpha parametr se použil pro evaluaci na testovacích datech. Výsledky pro klasifikaci vtipnosti i kategorie jsou zobrazeny v tabulce 7.2.

7.2 Trénování klasifikátoru

Při trénování klasifikátoru se ladí hyper-parametry popsané v sekci 5.1. Jelikož klasifikace vtipnosti i kategorie vychází z velmi podobného návrhu, hyper-parametry se nejprve ladí pro klasifikaci vtipnosti a poté se nejlepší řešení ověřují pro klasifikaci kategorií. Celkem bylo vyzkoušeno přes 20 různých kombinací parametrů ve snaze dosáhnout co nejlepšího řešení.

Nejprve se ladil počet epoch, tedy počet iterací přes celou trénovací sadu. Při experimentu s 10 epochami se ukázalo, že po páté epoše výrazně roste validační loss. Vyšší počet epoch způsobuje větší časovou náročnost, proto bylo pro zbytek experimentů zvoleno trénování s pěti epochami.

Kromě časové náročnosti je třeba řešit i náročnost paměťovou, která souvisí s velikostí batche. Experimenty probíhaly s velikostí batche 32, 16 a 8. Nakonec byla zvolena velikost 32, protože se s ní dosahovalo nejlepších výsledků s dostatečnou rychlostí trénování a přijatelnou paměťovou náročností.

Dalším z parametrů je počet batches po kterých se provádí evaluace. Provádění evaluace moc často je časově náročné, ale naopak méně častá evaluace přináší riziko minutí ideálního výsledku. Po několika experimentech se zvolila evaluace po 50 batches. Toto nastavení zaručuje dostatečně častou evaluaci s přijatelným časem trénování.

Dále se ladí maximální délka záznamu. Ta udává maximální počet tokenů jednoho vstupu. V datové sadě se nachází i velmi dlouhé záznamy, ale většinou se jedná o šum. Maximální počet tokenů pro BERT model je 512. Vyzkoušeny byly maximální velikosti 128, 256 a 512. Při zvýšení maximální délky z 128 na 256 dosahoval model výrazněji lepších výsledků. Zvýšení z 256 na 512 už zlepšení nepřineslo, a proto byla zvolena maximální velikost 256 tokenů.

Kromě délky záznamu je také možné nastavit, zda se bude vstup převádět na malé písmena nebo ne. Pro práci s *case sensitive* daty je třeba použít model *bert-base-cased*. Přihlížení na velikost znaků v textu nepřineslo zlepšení modelu a proto se používá převádění na malé písmena a model *bert-base-uncased*. Vyzkoušen byl i model *bert-large-uncased*, ale nedosahoval o tolik lepších výsledků.

Learning rate udává míru učení modelu. Pro AdamW optimalizátor se doporučují míry učení: $5E-5$, $3E-5$ a $2E-5$. Po experimentování se nejlépe osvědčila míra učení $5E-5$. Pro epsilon se doporučuje velmi malé číslo. Konkrétně byl zvolen jako $1E-8$.

Hlava klasifikátoru se skládá z dvouvrstvé architektury s ReLU. Velikost skrytých vrstev je nastavitelná, a proto se s ní také pracovalo během experimentů. Nejlepších výsledků model dosahoval pokud byla tato velikost nastavena na 50. Výstup první a vstup druhé vrstvy má tedy velikost 50. Experiment proběhl i s architekturou skládající se pouze z jedné vrstvy. Dvouvrstvá architektura dosahovala o něco lepších výsledků, ale v případě potřeby by pro menší náročnost bylo možné použít i jednovrstvou architekturu. Porovnání výsledků jednovrstvé a dvouvrstvé architektury pro klasifikaci vtipnosti je zobrazeno v tabulce 7.1.

	Parametr pro velikost skrytých vrstev	Přesnost	F1
Jednovrstvá architektura	-	0,7151	0,7232
Dvouvrstvá architektura	25	0,7185	0,7236
	50	0,7219	0,7274
	75	0,7021	0,7104

Tabulka 7.1: Porovnání výsledků modelu pro klasifikaci vtipnosti dle použité architektury hlavy klasifikátoru.

Výsledky experimentů

Výsledné řešení používá *bert-base-uncased* model, maximální délku vstupu 256, převádění vstupu na malé písmena, velikost batche 32 a počet epoch 5. Míra učení je nastavena na

$5E-5$, epsilon na $1E-8$ a pro velikost skrytých vrstev je nastavena hodnota 50. Klasifikátor rozpoznává 3 třídy pro vtipnost a 16 tříd pro kategorie. Evaluace se provádí po 50 batches.

Pro testovací sadu dosahuje klasifikátor u hodnocení vtipnosti přesnosti 0.72185 a F_1 0.72737. Pro odhad kategorie je přesnost 0.7055 a F_1 0.69491. Zlepšení oproti baseline je zobrazeno v tabulce 7.2. Hodnoty jsou průměrem výsledků ze tří běhů.

		Klasifikace vtipnosti	Klasifikace kategorie
Baseline	Přesnost	0.6583	0.4202
	F1	0.6455	0.3929
Klasifikátor	Přesnost	0.72185	0.7055
	F1	0.72737	0.69491

Tabulka 7.2: Porovnání výsledků baseline a vytvořeného klasifikátoru.

7.3 Porovnání vytvořeného klasifikátoru

Pro zjištění relevance řešení byl model použit na již řešený úkol rozpoznání humoru. Datová sada se nazývá *ColBERT* [2] a obsahuje 100,000 krátkých vtipů a stejný počet nevtipných novinových titulků. Datová sada je rozdělena na trénovací a testovací část. K datové sadě je vytvořen klasifikátor, který dosahuje přesnosti 0.982 a F_1 0.982.

Pro porovnání se oproti nastavení hyperparametrů z předchozí sekce snížila maximální délka vstupu na 128 tokenů, protože datová sada delší záznamy neobsahuje. Také se zvýšil počet batches po kterých se validuje na 500, protože datová sada je dost velká. Ze stejného důvodu se kvůli časové náročnosti iterovalo pouze přes 3 epochy. Ve třetí epoše už docházelo k přetrénování modelu, což vedlo k růstu validační loss.

S datovou sadu byly provedeny dva experimenty. První trénování používalo model, který dříve nebyl laděn na datové sadě vytvořené pro systém. Druhý pokus, který přinesl lepší výsledky, pro trénování na *ColBERT* datové sadě použil již laděný model popsaný výše v sekci 7.2. Výsledkem byla přesnost 0.986 a F_1 0.986. Porovnání výsledků pro je zobrazeno v tabulce 7.3.

	Použit dříve laděný model	Přesnost	F1
ColBERT	-	0.982	0.982
Vytvořený klasifikátor	Ne	0.985	0.985
	Ano	0.986	0.986

Tabulka 7.3: Porovnání výsledků *ColBERT* a vytvořeného klasifikátoru.

Kapitola 8

Závěr

Cílem této práce bylo vytvoření systému pro automatické hodnocení humoru. Nastudovaná teorie k humoru je včetně představení úkolu automatického hodnocení prezentována v kapitole 2. Kapitola se zaměřuje na tři dominantní teorie humoru a existující práce na téma počítačové klasifikace humoru. Jelikož tato klasifikace souvisí se zpracováním přirozeného jazyka a s hlubokým učením, kapitola 3 se těmto tématům věnuje detailněji.

Návrh systému je popsán v kapitole 4. Systém používá klasifikátor, který je schopný anglický vstupní text klasifikovat do třech úrovní vtipnosti a šestnácti kategorií. Pro uživatelské rozhraní je navržena webová aplikace. Webová aplikace je veřejně dostupná a kromě predikování nabízí i možnost poskytnout uživatelskou zpětnou vazbu. Implementace částí systému je popsána v kapitole 5.

Pro klasifikátor je použit již před-trénovaný BERT model. Model bylo třeba ale ještě doladit trénováním. K trénování jsou potřeba datové sady, kterým se věnuje kapitola 6. Nejprve kapitola představuje již existující datové sady související s hodnocením humoru. Poté popisuje vlastní datové sady, které se použily pro trénování klasifikátoru.

Kapitola 7 se věnuje experimentům. Převážně se jedná o ladění hyper-parametrů pro trénování modelu. K systému také byla vytvořena baseline pro porovnání s výsledným řešením. Výsledné řešení dosahuje podstatně lepších výsledků oproti baseline. Přesnost systému pro klasifikaci vtipnosti je 0.722. U klasifikace kategorie je přesnost 0.706. Řešení klasifikátoru je otestováno na již řešeném problému rozpoznání humoru. Systém zde dosáhl přesnosti 0.986, což je asi 0.4% zlepšení oproti řešení z porovnávané práce. Z výsledků lze vidět, že by systém mohl být nasazen pro tvůrce obsahu, kteří by si díky webové aplikaci mohli ověřovat vtipnost a kategorii svých textů.

Další vývoj, který by vedl k lepším výsledkům klasifikátoru, by mohl vycházet ze zvětšení datových sad pro trénování. Jelikož si systém uchovává predikce klasifikátoru a zpětné vazby od uživatelů, je možné model dále učit s těmito daty. Další možnosti je získání dalších externích dat, například opět pomocí web scrapingu. Pro klasifikaci kategorie je jedním z možných rozšíření predikce více kategorií pro jeden vstup, protože některé vtipy ze své podstaty spadají do více kategorií zároveň. Naopak pro hodnocení humoru by bylo zajímavé využít inkongruenci k predikování správné třídy.

Literatura

[1] Ahuja, V., Bali, T. a Singh, N. What makes us laugh? Investigations into Automatic Humor Classification. In: *Proceedings of the Second Workshop on Computational Modeling of People's Opinions, Personality, and Emotions in Social Media*. New Orleans, Louisiana, USA: Association for Computational Linguistics, červen 2018, s. 1–9. DOI: 10.18653/v1/W18-1101. Dostupné z: https://www.aclweb.org/anthology/W18-1101.

[2] Annamoradnejad, I. ColBERT: Using BERT Sentence Embedding for Humor Detection. *CoRR*. 2020, abs/2004.12765. Dostupné z: https://arxiv.org/abs/2004.12765.

[3] Aristotle, Crisp, R., Ameriks, K., Cambridge, U. of a Clarke, D. *Aristotle: Nicomachean Ethics*. Cambridge University Press, 2000. Cambridge Texts in the History of Philosophy. ISBN 9780521635462.

[4] ATTARDO, S. a RASKIN, V. Script theory revis(it)ed: joke similarity and joke representation model. *HUMOR*. Berlin, Boston: De Gruyter Mouton. 01 Jan. 1991, sv. 4, 3-4, s. 293 – 348. DOI: https://doi.org/10.1515/humr.1991.4.3-4.293. Dostupné z: https://www.degruyter.com/view/journals/humr/4/3-4/article-p293.xml.

[5] Beattie, J. *An Essay on Laughter and Ludicrous Composition, Written in the Year 1764*. W. Creech and E. and C. Dilly, 1776.

[6] Blum, E. K. The Heart of Computer Science. In: Blum, E. K. a Aho, A. V., ed. *Computer Science: The Hardware, Software and Heart of It*. New York, NY: Springer New York, 2011, s. 17–52. DOI: 10.1007/978-1-4614-1168-0_3. ISBN 978-1-4614-1168-0. Dostupné z: https://doi.org/10.1007/978-1-4614-1168-0_3.

[7] Cann, A., Calhoun, L. G. a Banks, J. S. On the role of humor appreciation in interpersonal attraction: It's no joking matter. *Humor*. WALTER DE GRUYTER & CO. 1997, sv. 10, s. 77–90.

[8] Chen, P.-Y. a Soo, V.-W. Humor Recognition Using Deep Learning. In: *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*. New Orleans, Louisiana: Association for Computational Linguistics, červen 2018, s. 113–117. DOI: 10.18653/v1/N18-2018. Dostupné z: https://www.aclweb.org/anthology/N18-2018.

[9] Chen, Z., Zhang, J. a Tao, D. Progressive lidar adaptation for road detection. *IEEE/CAA Journal of Automatica Sinica*. IEEE. 2019, sv. 6, č. 3, s. 693–702.

[10] Clark, A., Fox, C. a Lappin, S., ed. *The Handbook of Computational Linguistics and Natural Language Processing*. Wiley-Blackwell, 2010.

[11] Clevert, D.-A., Unterthiner, T. a Hochreiter, S. Fast and accurate deep network learning by exponential linear units (elus). *ArXiv preprint arXiv:1511.07289*. 2015.

[12] Dahl, G. E., Sainath, T. N. a Hinton, G. E. Improving deep neural networks for LVCSR using rectified linear units and dropout. In: *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*. 2013, s. 8609–8613. DOI: 10.1109/ICASSP.2013.6639346.

[13] Deng, J., Guo, J., Xue, N. a Zafeiriou, S. ArcFace: Additive Angular Margin Loss for Deep Face Recognition. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2019.

[14] Descartes, R. The Passions of the Soul. In: Cottingham, J., Stoothoff, R. a Murdoch, D., ed. *The Philosophical Writings of Descartes*. Cambridge University Press, 1985, sv. 1, s. 325–404. DOI: 10.1017/CBO9780511805042.010.

[15] Devlin, J., Chang, M.-W., Lee, K. a Toutanova, K. 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, červen 2019, s. 4171–4186. DOI: 10.18653/v1/N19-1423. Dostupné z: https://www.aclweb.org/anthology/N19-1423.

[16] Django Software Foundation. *Django*. Dostupné z: https://djangoproject.com.

[17] Engelthaler, T. a Hills, T. T. Humor norms for 4,997 English words. *Behavior research methods*. Springer. 2018, sv. 50, č. 3, s. 1116–1124.

[18] Epictetus. *The Enchiridion*. Project Gutenberg, 2004. ISBN 9780486111834.

[19] Eysenck, M. W. *Kognitivní psychologie*. Vyd. 1. Praha: Academia, 2008. ISBN 978-80-200-1559-4.

[20] Filatova, E. Irony and Sarcasm: Corpus Generation and Analysis Using Crowdsourcing. In: Citeseer. *Lrec*. 2012, s. 392–398.

[21] Fitch, F. B. Warren S. McCulloch and Walter Pitts. A logical calculus of the ideas immanent in nervous activity. Bulletin of mathematical biophysics, vol. 5 (1943), pp. 115–133. *Journal of Symbolic Logic*. Cambridge University Press. 1944, sv. 9, č. 2, s. 49–50. DOI: 10.2307/2268029.

[22] Foster, P. Model View Controller. *Beanz*. Červen 2019. Copyright - Copyright Owl Hill Media, LLC Jun 2019; Last updated - 2019-12-17. Dostupné z: https://search.proquest.com/magazines/model-view-controller/docview/2326508186/se-2?accountid=17115.

[23] Freud, S. *Jokes and their relation to the unconscious*. Routledge, 1960.

[24] Gibiansky, A., Arik, S. Ömer, Diamos, G. F., Miller, J., Peng, K. et al. Deep Voice 2: Multi-Speaker Neural Text-to-Speech. In: *NIPS*. 2017, s. 2962–2970. Dostupné z: http://www.papers.nips.cc/paper/6889-deep-voice-2-multi-speaker-neural-text-to-speech.

[25] Hobbes, T. Leviathan. In: Lawrence, B. B. a Karim, A., ed. *On Violence*. Duke University Press, 2007, s. 399–415. DOI: doi:10.1515/9780822390169-054. Dostupné z: https://doi.org/10.1515/9780822390169-054.

[26] Hochreiter, S. a Schmidhuber, J. Long Short-term Memory. *Neural computation*. Prosinec 1997, sv. 9, s. 1735–80. DOI: 10.1162/neco.1997.9.8.1735.

[27] Hossain, N., Krumm, J. a Gamon, M. "President Vows to Cut <Taxes> Hair": Dataset and Analysis of Creative Text Editing for Humorous Headlines. 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, červen 2019, s. 133–142. DOI: 10.18653/v1/N19-1012. Dostupné z: https://www.aclweb.org/anthology/N19-1012.

[28] Hossain, N., Krumm, J., Sajed, T. a Kautz, H. Stimulating Creativity with FunLines: A Case Study of Humor Generation in Headlines. In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*. Online: Association for Computational Linguistics, červenec 2020, s. 256–262. DOI: 10.18653/v1/2020.acl-demos.28. Dostupné z: https://www.aclweb.org/anthology/2020.acl-demos.28.

[29] Hutcheson, F. *Reflections Upon Laughter: And Remarks Upon the Fable of the Bees*. Garland Publishing, 1750.

[30] Ioffe, S. a Szegedy, C. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. In: Bach, F. a Blei, D., ed. *Proceedings of the 32nd International Conference on Machine Learning*. Lille, France: PMLR, 07–09 Jul 2015, sv. 37, s. 448–456. Proceedings of Machine Learning Research. Dostupné z: http://proceedings.mlr.press/v37/ioffe15.html.

[31] Jakubovitz, D., Giryes, R. a Rodrigues, M. R. D. Generalization Error in Deep Learning. In: Boche, H., Caire, G., Calderbank, R., Kutyniok, G., Mathar, R. et al., ed. *Compressed Sensing and Its Applications: Third International MATHEON Conference 2017*. Cham: Springer International Publishing, 2019, s. 153–193. DOI: 10.1007/978-3-319-73074-5_5. ISBN 978-3-319-73074-5. Dostupné z: https://doi.org/10.1007/978-3-319-73074-5_5.

[32] Jelinek, F., Mercer, R. L., Bahl, L. R. a Baker, J. K. Perplexity – a measure of the difficulty of speech recognition tasks. *Journal of the Acoustical Society of America*. November 1977, sv. 62, s. S63. Supplement 1.

[33] Jing, K. a Xu, J. A Survey on Neural Network Language Models. *CoRR*. 2019, abs/1906.03591. Dostupné z: http://arxiv.org/abs/1906.03591.

[34] Kant, I., Pluhar, W. a Gregor, M. *Critique of Judgment*. Hackett Publishing Company, Incorporated, 1987. Hackett Classics. ISBN 9781603846325.

[35] Khodak, M., Saunshi, N. a Vodrahalli, K. A Large Self-Annotated Corpus for Sarcasm. In: *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*. Miyazaki, Japan: European Language Resources Association (ELRA), Květen 2018. Dostupné z: https://www.aclweb.org/anthology/L18-1102.

[36] Kingma, D. P. a Ba, J. Adam: A Method for Stochastic Optimization. In: Bengio, Y. a LeCun, Y., ed. *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*. 2015. Dostupné z: http://arxiv.org/abs/1412.6980.

[37] Krizhevsky, A., Sutskever, I. a Hinton, G. ImageNet Classification with Deep Convolutional Neural Networks. *Neural Information Processing Systems*. Leden 2012, sv. 25. DOI: 10.1145/3065386.

[38] Krogh, A. a Hertz, J. A Simple Weight Decay Can Improve Generalization. In: Moody, J., Hanson, S. a Lippmann, R. P., ed. *Advances in Neural Information Processing Systems*. Morgan-Kaufmann, 1992, sv. 4. Dostupné z: https://proceedings.neurips.cc/paper/1991/file/8eefcfdf5990e441f0fb6f3fad709e21-Paper.pdf.

[39] Labutov, I. a Lipson, H. Humor as Circuits in Semantic Networks. In: *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. Jeju Island, Korea: Association for Computational Linguistics, červenec 2012, s. 150–155. Dostupné z: https://www.aclweb.org/anthology/P12-2030.

[40] Lembersky, G., Ordan, N. a Wintner, S. Language Models for Machine Translation: Original vs. Translated Texts. *Computational Linguistics*. 2012, sv. 38, č. 4, s. 799–825. DOI: 10.1162/COLI_a_00111. Dostupné z: https://www.aclweb.org/anthology/J12-4004.

[41] Liddy, E. D. Natural Language Processing. In: *Encyclopedia of Library and Information Science*. 2. vyd. New York: Marcel Dekker Inc, Květen 2003. ISBN 9780824742591.

[42] Loshchilov, I. a Hutter, F. Fixing Weight Decay Regularization in Adam. *CoRR*. 2017, abs/1711.05101. Dostupné z: http://arxiv.org/abs/1711.05101.

[43] Mareš, P. *Úvod do lingvistiky a lingvistické bohemistiky*. Karolinum Press, 2014. ISBN 9788024626406.

[44] McKinney Wes. Data Structures for Statistical Computing in Python. In: Walt Stéfan van der a Millman Jarrod, ed. *Proceedings of the 9th Python in Science Conference*. 2010, s. 56 – 61. DOI: 10.25080/Majora-92bf1922-00a.

[45] Medsker, L. a Jain, L. *Recurrent Neural Networks: Design and Applications*. CRC Press, 1999. International Series on Computational Intelligence. ISBN 9781420049176.

[46] Mei, H., Bansal, M. a Walter, M. Coherent Dialogue with Attention-Based Language Models. *Proceedings of the AAAI Conference on Artificial Intelligence*. Feb. 2017, sv. 31, č. 1. Dostupné z: https://ojs.aaai.org/index.php/AAAI/article/view/10961.

[47] Mihalcea, R. a Strapparava, C. Making Computers Laugh: Investigations in Automatic Humor Recognition. In: *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*. Vancouver, British Columbia, Canada: Association for Computational Linguistics, říjen 2005, s. 531–538. Dostupné z: https://www.aclweb.org/anthology/H05-1067.

[48] Mikolov, T., Chen, K., Corrado, G. a Dean, J. Efficient Estimation of Word Representations in Vector Space. In: Bengio, Y. a LeCun, Y., ed. *1st International Conference on Learning Representations, ICLR 2013, Scottsdale, Arizona, USA, May 2-4, 2013, Workshop Track Proceedings*. 2013. Dostupné z: http://arxiv.org/abs/1301.3781.

[49] Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S. a Dean, J. Distributed Representations of Words and Phrases and their Compositionality. In: Burges, C. J. C., Bottou, L., Welling, M., Ghahramani, Z. a Weinberger, K. Q., ed. *Advances in Neural Information Processing Systems*. Curran Associates, Inc., 2013, sv. 26, s. 3111–3119. Dostupné z: https://proceedings.neurips.cc/paper/2013/file/9aa42b31882ec039965f3c4923ce901b-Paper.pdf.

[50] Mikolov, T. Language models for automatic speech recognition of Czech lectures. In: *Proc. STUDENT EEICT 2008*. Faculty of Electrical Engineering and Communication BUT, 2008, s. 1–5. ISBN 978-80-214-3617-6. Dostupné z: https://www.fit.vut.cz/research/publication/8749.

[51] Mikolov, T., Karafiát, M., Burget, L., Černocký, J. a Khudanpur, S. Recurrent neural network based language model. In: *Proceedings of the 11th Annual Conference of the International Speech Communication Association (INTERSPEECH 2010)*. International Speech Communication Association, 2010, sv. 2010, č. 9, s. 1045–1048. ISBN 978-1-61782-123-3. Dostupné z: https://www.fit.vut.cz/research/publication/9362.

[52] Miller, T., Hempelmann, C. a Gurevych, I. SemEval-2017 Task 7: Detection and Interpretation of English Puns. In: *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*. Vancouver, Canada: Association for Computational Linguistics, Srpen 2017, s. 58–68. DOI: 10.18653/v1/S17-2005. Dostupné z: https://www.aclweb.org/anthology/S17-2005.

[53] Miller, T., Hempelmann, C. F. a Gurevych, I. SemEval-2017 Task 7: Detection and Interpretation of English Puns. In: *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*. Srpen 2017, s. 59–69. ISBN 978-1-945626-00-5.

[54] Morreall, J. *The Philosophy of Laughter and Humor*. State University of New York Press, 1987. SUNY series in philosophy. ISBN 9780887063275.

[55] Morreall, J. Philosophy of Humor. In: Zalta, E. N., ed. *The Stanford Encyclopedia of Philosophy* [https://plato.stanford.edu/archives/fall2020/entries/humor/]. Fall 2020. Metaphysics Research Lab, Stanford University, 2020.

[56] Nayak, P. *Understanding searches better than ever before* [online]. 2019 [cit. 2020-07-07]. Dostupné z: https://www.blog.google/products/search/search-language-understanding-bert/.

[57] Nijholt, A., Stock, O., Dix, A. a Morkes, J. Humor Modeling in the Interface. In: *CHI '03 Extended Abstracts on Human Factors in Computing Systems*. New York, NY, USA: Association for Computing Machinery, 2003, s. 1050–1051. CHI EA '03. DOI: 10.1145/765891.766143. ISBN 1581136374. Dostupné z: https://doi.org/10.1145/765891.766143.

[58] Pala, K. *Počítačové zpracování přirozeného jazyka*. 1. Brno: FI MU, 2000.

[59] Pascanu, R., Mikolov, T. a Bengio, Y. On the difficulty of training recurrent neural networks. In: Dasgupta, S. a McAllester, D., ed. *Proceedings of the 30th International Conference on Machine Learning*. Atlanta, Georgia, USA: PMLR, 17–19 Jun 2013, sv. 28, č. 3, s. 1310–1318. Proceedings of Machine Learning Research. Dostupné z: http://proceedings.mlr.press/v28/pascanu13.html.

[60] Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J. et al. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In: Wallach, H., Larochelle, H., Beygelzimer, A., Alché Buc, F. d', Fox, E. et al., ed. *Advances in Neural Information Processing Systems 32*. Curran Associates, Inc., 2019, s. 8024–8035. Dostupné z: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.

[61] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B. et al. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*. 2011, sv. 12, s. 2825–2830.

[62] Pennington, J., Socher, R. a Manning, C. 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, říjen 2014, s. 1532–1543. DOI: 10.3115/v1/D14-1162. Dostupné z: https://www.aclweb.org/anthology/D14-1162.

[63] Peters, M., Neumann, M., Iyyer, M., Gardner, M., Clark, C. et al. Deep Contextualized Word Representations. In: *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*. New Orleans, Louisiana: Association for Computational Linguistics, červen 2018, s. 2227–2237. DOI: 10.18653/v1/N18-1202. Dostupné z: https://www.aclweb.org/anthology/N18-1202.

[64] Plato, H. Philebus. In: Cairns, H., ed. *The Collected Dialogues of Plato*. Princeton University Press, 1961, s. 1086–1150. DOI: doi:10.1515/9781400835867-025. Dostupné z: https://doi.org/10.1515/9781400835867-025.

[65] Plato, H. Republic. In: Cairns, H., ed. *The Collected Dialogues of Plato*. Princeton University Press, 1961, s. 575–844. DOI: doi:10.1515/9781400835867-020. Dostupné z: https://doi.org/10.1515/9781400835867-020.

[66] Potash, P., Romanov, A. a Rumshisky, A. SemEval-2017 Task 6: #HashtagWars: Learning a Sense of Humor. In: *Proceedings of the 11th International Workshop on*

Semantic Evaluation (SemEval-2017). Vancouver, Canada: Association for Computational Linguistics, Srpen 2017, s. 49–57. DOI: 10.18653/v1/S17-2004. Dostupné z: https://www.aclweb.org/anthology/S17-2004.

[67] Pungas, T. *A dataset of English plaintext jokes*. GitHub, 2017. Dostupné z: https://github.com/taivop/joke-dataset.

[68] Radford, A., Narasimhan, K., Salimans, T. a Sutskever, I. Improving language understanding by generative pre-training. 2018. Dostupné z: https://www.cs.ubc.ca/~amuham01/LING530/papers/radford2018improving.pdf.

[69] Rajalakshmi, Madeshan, Narayanan a Scholar, U. An Exclusive Study on Unstructured Data Mining with Big Data. Leden 2015, sv. 10, s. 973–45623875.

[70] Raschka, S. Naive Bayes and Text Classification I - Introduction and Theory. *CoRR*. 2014, abs/1410.5329. Dostupné z: http://arxiv.org/abs/1410.5329.

[71] Raskin, V. Semantic Mechanisms of Humor. *Annual Meeting of the Berkeley Linguistics Society*. 1979, sv. 5, s. 325. ISSN 0363-2946.

[72] Reddi, S. J., Kale, S. a Kumar, S. On the Convergence of Adam and Beyond. *CoRR*. 2019, abs/1904.09237. Dostupné z: http://arxiv.org/abs/1904.09237.

[73] Ruck, D. W., Rogers, S. K. a Kabrisky, M. Feature selection using a multilayer perceptron. *Journal of Neural Network Computing*. 1990, sv. 2, č. 2, s. 40–48.

[74] Salazar, J., Liang, D., Nguyen, T. Q. a Kirchhoff, K. Masked Language Model Scoring. In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Online: Association for Computational Linguistics, červenec 2020, s. 2699–2712. DOI: 10.18653/v1/2020.acl-main.240. Dostupné z: https://www.aclweb.org/anthology/2020.acl-main.240.

[75] Schopenhauer, A., Haldane, R. a Kemp, J. *The World as Will and Idea*. Library of Alexandria, 2020. Library of Alexandria. ISBN 9781465553515.

[76] Shaftesbury, A. *Sensus Communis: An Essay on the Freedom of Wit and Humour. In a Letter to a Friend*. E. Sanger, 1709.

[77] Shiloh-Perl, L. a Giryes, R. Introduction to deep learning. *CoRR*. 2020, abs/2003.03253. Dostupné z: https://arxiv.org/abs/2003.03253.

[78] Shorten, C. a Khoshgoftaar, T. A survey on Image Data Augmentation for Deep Learning. *Journal of Big Data*. Červenec 2019, sv. 6. DOI: 10.1186/s40537-019-0197-0.

[79] Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I. a Salakhutdinov, R. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research*. 2014, sv. 15, č. 56, s. 1929–1958. Dostupné z: http://jmlr.org/papers/v15/srivastava14a.html.

[80] Stock, O. a Strapparava, C. Laughing with HAHAcronym, a Computational Humor System. In: *Proceedings of the 21st National Conference on Artificial Intelligence - Volume 2*. AAAI Press, 2006, s. 1675–1678. AAAI'06. ISBN 9781577352815.

[81] Tecuci, G. Artificial intelligence. *Wiley Interdisciplinary Reviews: Computational Statistics*. Hoboken, USA: John Wiley & Sons, Inc. 2012, sv. 4, č. 2, s. 168–180. ISSN 1939-5108.

[82] Tiedemann, J. Parallel Data, Tools and Interfaces in OPUS. In: Chair), N. C. C., Choukri, K., Declerck, T., Dogan, M. U., Maegaard, B. et al., ed. *Proceedings of the Eight International Conference on Language Resources and Evaluation (LREC'12)*. Istanbul, Turkey: European Language Resources Association (ELRA), May 2012. ISBN 978-2-9517408-7-7.

[83] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L. et al. Attention is All you Need. In: Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R. et al., ed. *Advances in Neural Information Processing Systems*. Curran Associates, Inc., 2017, sv. 30. Dostupné z: https://proceedings.neurips.cc/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.

[84] Vilalta, R., Giraud Carrier, C., Brazdil, P. a Soares, C. Inductive Transfer. In: Sammut, C. a Webb, G. I., ed. *Encyclopedia of Machine Learning*. Boston, MA: Springer US, 2010, s. 545–548. DOI: 10.1007/978-0-387-30164-8_401. ISBN 978-0-387-30164-8. Dostupné z: https://doi.org/10.1007/978-0-387-30164-8_401.

[85] Weller, O. a Seppi, K. Humor Detection: A Transformer Gets the Last Laugh. In: *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. Hong Kong, China: Association for Computational Linguistics, Listopad 2019, s. 3621–3625. DOI: 10.18653/v1/D19-1372. Dostupné z: https://www.aclweb.org/anthology/D19-1372.

[86] Wilbur, C. J. a Campbell, L. Humor in Romantic Contexts: Do Men Participate and Women Evaluate? *Personality and Social Psychology Bulletin*. 2011, sv. 37, č. 7, s. 918–929. DOI: 10.1177/0146167211405343. PMID: 21521721. Dostupné z: https://doi.org/10.1177/0146167211405343.

[87] Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C. et al. Transformers: State-of-the-Art Natural Language Processing. In: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*. Online: Association for Computational Linguistics, říjen 2020, s. 38–45. Dostupné z: https://www.aclweb.org/anthology/2020.emnlp-demos.6.

[88] Xu, B., Wang, N., Chen, T. a Li, M. Empirical Evaluation of Rectified Activations in Convolutional Network. *CoRR*. 2015, abs/1505.00853. Dostupné z: http://arxiv.org/abs/1505.00853.

[89] Yu, Z., Tan, J. a Wan, X. A Neural Approach to Pun Generation. In: *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Melbourne, Australia: Association for Computational Linguistics, červenec 2018, s. 1650–1660. DOI: 10.18653/v1/P18-1153. Dostupné z: https://www.aclweb.org/anthology/P18-1153.

Příloha A

Obsah přiloženého paměťového média

- **ahaha**: zdrojové soubory pro webovou aplikaci a klasifikaci
- **docs**: dokumentace zdrojových souborů práce
- **src-latex**: zdrojové soubory textu práce
- **automaticke-hodnoceni-humoru.pdf**: text práce ve formátu PDF