VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky
a komunikačních technologií

DIPLOMOVÁ PRÁCE

Brno, 2020 Bc. Yehor Safonov

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

ÚSTAV TELEKOMUNIKACÍ
DEPARTMENT OF TELECOMMUNICATIONS

FILTROVÁNÍ SPAMOVÝCH ZPRÁV POMOCÍ METOD UMĚLÉ INTELIGENCE
EMAIL SPAM FILTERING USING ARTIFICIAL INTELLIGENCE

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — **Bc. Yehor Safonov**
AUTHOR

VEDOUCÍ PRÁCE — **Ing. Martin Kolařík**
SUPERVISOR

BRNO 2020

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Diplomová práce

magisterský navazující studijní obor **Informační bezpečnost**

Ústav telekomunikací

Student: Bc. Yehor Safonov ***ID:*** 185942
Ročník: 2 ***Akademický rok:*** 2019/20

NÁZEV TÉMATU:

Filtrování spamových zpráv pomocí metod umělé inteligence

POKYNY PRO VYPRACOVÁNÍ:

Nastudujte současné metody filtrování spamových zpráv, které používají tzv. hluboké učení. V rámci diplomové práce proveďte rešerši těchto metod a jejich vlastností se zaměřením na použití pro filtrování spamových zpráv. V rámci teoretické části vysvětlete problematiku e-mailové komunikace se zaměřením na nevyžádanou poštu. Popište nejběžnější spamové techniky a porovnejte existující způsoby ochrany. V rámci praktické části charakterizujte vybranou datovou sadu a popište proces její zpracování. Dále ze zkoumaných technik hlubokého učení vyberte vhodné metody a aplikujte je na datovou sadu e-mailů. Výběr zdůvodněte a metody otestujte pro použití na daný problém. Metody porovnejte na základě naměřené přesnosti. Výsledky testování vhodně prezentujte a diskutujte možná vylepšení.

DOPORUČENÁ LITERATURA:

[1] BHOWMICK, Alexy; HAZARIKA, Shyamanta M. E-mail spam filtering: A review of techniques and trends. In: Advances in Electronics, Communication and Computing. Springer, Singapore, 2018. p. 583-590.

[2] DADA, Emmanuel Gbenga, et al. Machine learning for email spam filtering: review, approaches and open research problems. Heliyon, 2019, 5.6: e01802.

Termín zadání: 3.2.2020 ***Termín odevzdání:*** 1.6.2020

Vedoucí práce: Ing. Martin Kolařík

prof. Ing. Jiří Mišurec, CSc.
předseda oborové rady

ABSTRAKT

V moderním počítačovém světě e-mailová komunikace patří do nejpoužívanějších prostředků pro výměnu zpráv mezi uživateli. Jedná se o volně dostupný, efektivní a jednoduchý způsob sloužící ke sdělení informací. Tyto tři základní pilíře přispívají k její světové rozšířenosti a strmému nárůstu přenášených elektronických zpráv. Na druhou stranu, rostoucí popularita této technologie v sobě skrývá velká bezpečnostní rizika a tvoří z ní ideální nástroj pro šíření nevyžádaného obsahu a realizaci útoků cílených jak na koncové uživatele, tak i na celé počítačové infrastruktury. Ačkoliv v dnešní době používané klasické nástroje na filtrování spamu dosahují vysokých přesností, často neumožňují pokrytí dynamičnosti vývoje spamových technik a trpí problémy s přeučením, uváznutím v nevhodných lokálních minimech, neschopností efektivně zpracovávat vysoce dimenzionální data a z dlouhodobého hlediska disponují problémy s udržitelností. Hlavním cílem této diplomové práce je vytvoření a naučení modelů hlubokých neuronových sítí použitím nejmodernějších technik a přístupů existujících ve světě zpracování přirozeného jazyka a strojového učení. V rámci teoretické části se práce zaměřuje na problematiku e-mailové komunikace se zaměřením na filtrování nevyžádané pošty. Následně se věnuje doméně strojového učení a umělých neuronových sítí, zejména principům jejich fungování, základním vlastnostem a možnostem jejich aplikování na okruh problémů spojených s provedením textové analýzy. Mezi silné stránky práce patří provedení podrobného srovnání současných metod strojového učení, jejich specifik a přesnosti při aplikování na klasifikaci spamu. V praktické části práce byl důraz položen na zpracování datové sady surových e-mailů a srovnání modelů ULMFiT, BERT a XLNet. Zpracování dat bylo rozděleno do pěti etap, a to s cílem zachování co nejvyšší informační hodnoty zpráv a vytvoření kvalitní datové sady, která byla použita pro trénování, testování a validaci zvolených druhů neuronových sítí. Dále diplomová práce zahrnuje popis procesu učení sítí včetně etapy finálního přizpůsobení dat k modelování. Na konci práce byly implementované modely srovnány a byla nastíněna případná rozšíření do budoucna.

KLÍČOVÁ SLOVA

BERT, bezpečnost, e-mailová komunikace, filtrování spamu, hluboké učení, textová klasifikace, ULMFiT, umělá inteligence, zpracování přirozeného jazyka, XLNet.

ABSTRACT

In the modern world, email communication defines itself as the most used technology for exchanging messages between users. It is based on three pillars which contribute to the popularity and stimulate its rapid growth. These pillars are represented by free availability, efficiency and intuitiveness during exchange of information. All of them constitute a significant advantage in the provision of communication services. On the other hand, the growing popularity of email technologies poses considerable security risks and transforms them into an universal tool for spreading unsolicited content. Potential attacks may be aimed at either a specific endpoints or whole computer infrastructures. Despite achieving high accuracy during spam filtering, traditional techniques do not often catch up to rapid growth and evolution of spam techniques. These approaches are affected by overfitting issues, converging into a poor local minimum, inefficiency in highdimensional data processing and have long-term maintainability issues. One of the main goals of this master's thesis is to develop and train deep neural networks using the latest machine learning techniques for successfully solving text-based spam classification problem belonging to the Natural Language Processing (NLP) domain. From a theoretical point of view, the master's thesis is focused on the e-mail communication area with an emphasis on spam filtering. Next parts of the thesis bring attention to the domain of machine learning and artificial neural networks, discuss principles of their operations and basic properties. The theoretical part also covers possible ways of applying described techniques to the area of text analysis and solving NLP. One of the key aspects of the study lies in a detailed comparison of current machine learning methods, their specifics and accuracy when applied to spam filtering. At the beginning of the practical part, focus will be placed on the e-mail dataset processing. This phase was divided into five stages with the motivation of maintaining key features of the raw data and increasing the final quality of the dataset. The created dataset was used for training, testing and validation of types of the chosen neural networks. Selected models ULMFiT, BERT and XLNet have been successfully implemented. The master's thesis includes a description of the final data adaptation, neural networks learning process, their testing and validation. In the end of the work, the implemented models are compared using a confusion matrix and possible improvements and concise conclusion are also outlined.

KEYWORDS

Artificial intelligence, BERT, deep learning, email communication, natural language processing, security, spam filtering, text classification, ULMFiT, XLNet.

SAFONOV, Yehor. *Filtrování spamových zpráv pomocí metod umělé inteligence*. Brno, 2019, 150 s. Diplomová práce. Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací. Vedoucí práce: Ing. Martin Kolařík

PROHLÁŠENÍ

Prohlašuji, že svou diplomovou práci na téma „Filtrování spamových zpráv pomocí metod umělé inteligence“ jsem vypracoval samostatně pod vedením vedoucího diplomové práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedené diplomové práce dále prohlašuji, že v souvislosti s vytvořením této diplomové práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a/nebo majetkových a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon), ve znění pozdějších předpisů, včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

Brno ....

podpis autora

PODĚKOVÁNÍ

Na tomto místě bych chtěl nechat slova vděčnosti všem lidem, jež jsem měl štěstí během studia na vysoké škole potkat a kteří mi pomohli vybudovat kapitál zkušeností v oblasti informačních technologií a počítačové bezpečnosti. Rád bych poděkoval vedoucímu diplomové práce panu Ing. Martinu Kolaříkovi a také technickému konzultantu doc. Ing. Radimu Burgetovi, Ph.D. za odborné vedení, konzultace, trpělivost a podnětné návrhy k práci. Zároveň bych chtěl poděkovat společnostem TrustPort a.s. a AEC a.s., obzvlášť Ing. Romanu Vencálkovi za vstřícnost a praktická doporučení v oblasti zpracování nevyžádané pošty. V neposlední řádě bych chtěl zmínit svou rodinu a kamarády, kteří mě vždy psychicky podporovali a motivovali dosáhnout lepších výsledků.

Obsah

Seznam obrázků

Seznam tabulek

Úvod

Elektronická pošta (zkráceně e-mail) je technologií sloužící pro výměnu zpráv mezi uživateli v internetovém světě [1, 2]. Jedná se o efektivní způsob elektronické komunikace, který nevyžaduje od uživatele prakticky žádnou peněžní investici. Je možné říci, že existence a volná dostupnost elektronické pošty poskytly neuvěřitelný prostor pro rozvoj v oblasti poskytování komunikačních služeb. Nezanedbatelnou výhodou e-mailové komunikace je její cena, rychlost a jednoduchost [3]. Ve své podstatě pro úspěšné založení e-mailové schránky a zasílání e-mailových zpráv stačí disponovat jednoduchým výpočetním zařízením a mít připojení k internetu [2]. Pro pochopení rozsáhlosti a významnosti této technologie je důležité zmínit, že na konci roku 2019 celkový denní počet uživatelských zpráv převýšil 293 miliardy a celkový počet uživatelů přesáhl hranici 4,3 miliardy, což činí přibližně 55,8 % obyvatel naší planety [4].

Popularita elektronické komunikace přináší s sebou i bezpečnostní rizika specifická pro tyto technologie [5]. Denně se v e-mailových schránkách uživatelů vyskytuje nevyžádaná a nechtěná pošta (*spam*), která za sebou může skrývat snahu útočníka získat citlivé údaje oběti, poskytnout nechtěné služby a zboží, rozšířit škodlivý kód apod. K dalším problémům je možné přiřadit neoprávněné přečtení a modifikaci obsahu zpráv, podvržení identity odesílatele, odepření služeb atd. Podle statistické analýzy činí celkový denní objem nevyžádaných e-mailových zpráv přibližně 165 miliardy, což odpovídá 56,5 % celkového počtu e-mailů [6, 7]. Je celkem zřejmé, že rozšířenost elektronické pošty v kombinaci s nízkými náklady vynaloženými na zasílání e-mailů ji dělají ideálním nástrojem na uskutečnění pokročilých útoků. Predikovatelné také je, že v budoucnu popularita nevyžádané pošty bude růst, a proto otázky spojené se zajištěním bezpečnosti a filtrováním nedůvěryhodných e-mailů není možné podceňovat a odkládat na vedlejší kolej [1, 5].

Jedním z hlavních cílů této diplomové práce je vytvořit takový nástroj, který by dokázal co nejpřesněji a co nejefektivněji provádět klasifikaci příchozí elektronické pošty, rozpoznávat a vyfiltrovávat nevyžádané zprávy, předcházet vzniku negativních dopadů hrozeb cílených na koncové uživatele anebo celé počítačové infrastruktury. Výhodou a zároveň unikátní vlastností vytvořeného nástroje bude aplikování nejmodernějších způsobů klasifikace, tj. *state-of-the-art* metod umělé inteligence sloužící k rozpoznání textu. Další silnou stránkou naprogramovaného řešení bude jeho preciznost (nízké množství *false positives* a *false negatives*) z důvodu použití rozsáhlé datové sady různorodých nevyžádaných e-mailů. Navíc je plánováno vytvořit sadu skriptů umožňující zajistit *end-to-end* zpracování e-mailů a docílit takového stavu, aby bylo možné příchozí elektronické zprávy bezproblémově zpracovat a transformovat do podoby, která by umožnila transparentní napojení na vstupní vrstvy neuronové sítě a jejich následující klasifikaci.

Diplomová práce je rozdělena na čtyři kapitoly, z nichž každá navazuje na předchozí a určitým způsobem ji doplňuje a rozšiřuje. V prvních dvou kapitolách je důraz kladen na teoretické vysvětlení probírané problematiky. Kapitoly se věnují vysvětlení základních specifik e-mailové komunikace a její bezpečnosti. Následně dochází k popisu klíčových vlastností neuronových sítí a porovnání existujících modelů umělé inteligence. Popsaná teoretická báze by měla posloužit jako znalostní kapitál pro úspěšné dokončení praktických částí této diplomové práce. Realizační část tvoří hlavní jádro diplomové práce, v rámci níž dochází k realizaci získaných znalostí, zpracování datové sady, vytvoření a otestování různých modelů neuronových sítí. Finální struktura diplomové práce je uvedena a stručně popsána níže.

V první kapitole bude důraz kladen na problematiku e-mailové komunikace. V rámci této kapitoly bude popsána struktura a formáty e-mailových zpráv. Následně bude vysvětlen princip předání elektronických zpráv v rámci internetu a budou popsány protokoly umožňující provedení této komunikace. Druhá část této kapitoly bude věnována hlavně problematice nevyžádané pošty, tj. spamu. V rámci této části budou uvedeny statistiky a provedena analýza nevyžádaných zpráv. V textu budou rozebrány a podrobně vysvětleny různé druhy spamu a možné vektory útoků, jejichž znalost je pro úspěšnou ochranu koncových uživatelů nezbytná. Na konci této části budou uvedeny existující způsoby ochrany proti spamu, vysvětlen princip jejich fungování a zmíněny silné a slabé stránky každého z nich.

Druhá kapitola se zabývá problematikou umělé inteligence. Konkrétně popisuje základní vlastnosti neuronových sítí, existující architektury a stavební prvky. V rámci této kapitoly budou vysvětleny klíčové pojmy spojené s neuronovými sítěmi a procesem jejich učení. V dalších krocích dojde k vysvětlení principů zpracování a rozpoznání přirozeného jazyka, budou popsány a porovnány moderní metody založené na hlubokém učení. Vytvořený přehled nejmodernějších hlubokých neuronových sítí poslouží jako základ pro provedení výběru konkrétních modelů, jejichž inicializace, trénování a testování bude provedeno v praktické části diplomové práce. K hlavním cílům druhé kapitoly lze také zařadit vytvoření kompletního přehledu existujících technik strojového učení, které se v současné době používají v bezpečnostních mechanismech na filtrování nevyžádaného obsahu. Kromě základního popisu existujících technik budou do popisu zahrnuty jejich specifika, silné a slabé stránky. Navíc, v souladu se zadáním diplomové práce, bude důraz položen na techniky hlubokého učení. V rámci daného popisu dojde k provedení rozsáhlé rešerše vědeckých prací s cílem vytvoření základní křivky úspěšnosti existujících algoritmů sloužících ke klasifikaci spamu. Ve finále budou vybrány nejaktuálnější vědecké články, které jsou nejvíce porovnatelné s modely realizovanými v dané diplomové práci. Každá vybraná studie bude stručně popsána a její klíčové vlastnosti budou přehledně seřazeny na konci kapitoly do závěrečné tabulky.

Třetí kapitola diplomové práce tvoří jádro diplomové práce a zaměřuje se na proces tvorby funkčních modelů neuronových sítí. Pozornost je věnována vytvoření agregované datové sady vhodné pro učení, která by umožňovala vytvořit co nejpřesnější modely s nízkou mírou falešných detekcí. V první řadě zde dochází k popisu dílčích datových sad a vytvoření jejich detailních charakteristik. Následně se práce věnuje tvorbě programu, neboli parseru, jehož primárním účelem bude předzpracování surových e-mailů, oprava chyb, jejich vytřídění a převedení do unifikovaného formátu. Vytvoření parseru umožní docílit plné automatizace procesu vyhodnocení příchozích e-mailů a v reálném čase provádět jejich předzpracování, napojení na neuronovou síť a klasifikaci. Po předzpracování parserem bude vytvořená datová sada e-mailů rozdělena na tři speciálně vybrané části, které splňují podmínky definované po pokusném natrénování jednoho modelu sítě. První a největší část datové sady bude použita pro učení modelů. Druhá část zpracovaných e-mailů poslouží k provedení validace a vyhodnocení natrénovaných modelů. Třetí část datové sady bude zpracována zvlášť a poslouží k odhadu finálních přesností modelů v podmínkách, které jsou co nejvíce přiblížené k reálným. Další část třetí kapitoly je zaměřena na aplikování vybraných modelů neuronových sítí na datovou sadu pečlivě zpracovanou v rámci předchozích kroků. V rámci této kapitoly dojde k výběru tří modelů pokročilých neuronových sítí, kde každý z nich bude detailně nastudován. V rámci jednotlivých studií bude vytvořena kompletní charakteristika každého modelu, včetně popisu jeho architektury, použitých mechanismů sloužících k odhalení sémantických vazeb atd. Po uvedení výhod zvolených modelů bude přistoupeno k jejich implementaci. Tyto kroky budou zahrnovat finální přizpůsobení datové sady k modelování, inicializaci neuronových sítí, tokenizaci a numerikalizaci vstupních dat, napojení trénovacího a validačního datového rámce na vstup do modelů a na závěr – aplikování technik přeneseného učení s účelem vytvoření klasifikátoru spamu. Po ukončení procesu trénování budou na konci příslušných podkapitol zveřejněny hodnoty ztrátových funkcí modelů a také finální přesnosti na validačních datech.

Čtvrtá část diplomové práce se zaměřuje na závěrečné vyhodnocení modelů. Hlavním cílem této kapitoly bude určit sledované parametry, provést a popsat proces testování naučených modelů. Samotný proces testování bude proveden na odlišné datové množině, a to takové, která nebyla použitá během procesu učení. Z výše uvedeného důvodu bude na vyhodnocení sítí použit zvláštní testovací datový rámec. Přímé srovnání modelů bude provedeno pomocí matic záměn, celkové doby učení, finálních přesností, hodnot preciznosti, senzitivity, specificity a F-míry, podrobněji viz příslušná kapitola. Zároveň v rámci této kapitoly budou detailně vysvětleny problémy vzniklé během procesu učení a způsoby jejich vyřešení. Na konci čtvrté kapitoly budou popsána rozšíření, která jsou naplánována do budoucna.

1 Problematika e-mailové komunikace

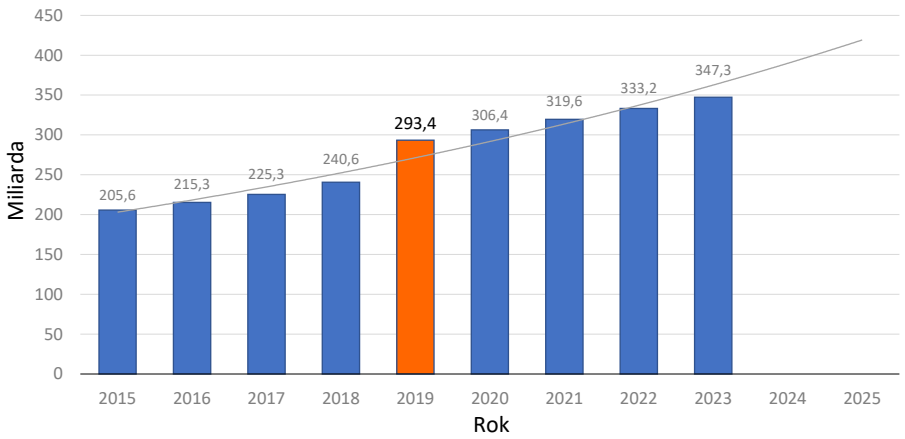
Slovo komunikace pochází z latinského *communicatio*, co v doslovném překladu znamená vyměňovat a také spojovat [8]. Tento pojem je velice široký a v nejširším slova smyslu je definován jako přenos informace, která má za cíl zmenšit míru neurčitosti komunikujících prvků [9]. V sociálním světě, tzn. v komunikaci sociální se jedná o jakousi formu sociální interakce, během které dochází k vzájemné stimulaci a sdílení informací [8]. Proces úspěšné komunikace není možný bez využití komunikačního kanálu, sdílené a uznávané na obou stranách abecedy symbolů apod. [9] Je důležité zmínit, že k procesu komunikace dochází vždy, kdy probíhá vysílání a příjem sdělení, a to i v případě, kdy sdělení není přímo vyžádáno anebo dokonce nechtěno [8].

Technologický pokrok rozhodně nestojí na místě, ba se neustále posouvá dopředu. Denně vznikají a zanikají technologie a mechanismy, jejichž existence přímo anebo nepřímo ovlivňuje procesy v sociálním světě a mění běžné vnímání světa. Stejně tak vznik[1] veřejně dostupné elektronické pošty v roce 1996, změnil způsob moderní komunikace. K používaným a hojně populárním v dvacátém století telefonním hovorům a ručně psáným dopisům se přidala elektronická pošta, neboli e-mail, který v nepopiratelné míře ovlivnil a změnil podobu moderní komunikace [1, 10]. Otázky ceny, vzdálenosti a dostupnosti odešly na vedlejší kolej, což přispělo k globální popularizaci této technologie. Spektrum používání e-mailové komunikace je velmi široké [1]. S elektronickými zprávami je možné se potkat jak v soukromém, tak i veřejném sektoru. Jedná se například o soukromou komunikaci mezi osobami, komunikaci v rámci firemního prostředí (interní a externí), komunikaci s orgány veřejné moci atd.

Na diagramu číslo 1.1 je znázorněna globální statistika e-mailových zpráv odeslaných za jeden den, a to včetně soukromého a veřejného sektoru. Zároveň na grafu je vidět předpověď na budoucích pět let [4]. Z uvedených statistik je patrné, že se každoročně zvyšuje popularita a zároveň denní přírůst zpráv. Jak bylo uvedeno v úvodu této práce celkový počet uživatelů v roce 2019 přesahl hranici v 4,3 miliardy a denní počet odeslaných zpráv převýšil 293 miliardy [4]. Jednou z příčin nárůstu e-mailového trafiku je možné zahlédnout v zvýšení celkového počtu e-mailových schránek (uživatele preferují mít několik zvláštních schránek na různé účely) [4].

Je důležité zmínit, že při svém vzniku byla otázka bezpečnosti e-mailové komunikace odložena na vedlejší kolej [5]. Z takovéhoto důvodu jsou i v dnešní době s problematikou e-mail komunikace úzce spojeny následující problémy: problematická

[1]Datem vzniku prvních elektronických zpráv je nejčastěji považován rok 1965, tzn. ještě před vznikem sítě *ARPANET* v roce 1969. Tyto zprávy byly používány výhradně pro uskutečnění komunikace mezi různými uživateli sálových počítačů [1, 2]. Každopádně k veřejnému rozšíření elektronické komunikace došlo až po vytvoření v roce 1996 první volné e-mailové služby *Hotmail*.



Obr. 1.1: Globální statistika e-mailových zpráv odeslaných za jeden den[2]

identifikace komunikujících stran, složitost zajištění integrity, důvěrnosti a nepopiratelnosti [1, 5]. Samozřejmě, aplikováním moderních kryptografických primitiv je možné docílit vyřešení všech výše zmíněných problémů, avšak je nutné při používání daných služeb výše zmíněnou skutečnost mít na vědomí.

Nakonec, při přímém porovnání elektronických e-mailů s klasickými prostředky doručování písemností je také vhodné zohlednit právní úpravu a možnost použití daného způsobu komunikace s orgány státního sektoru [12]. České právní normy bohužel neupřednostňují elektronických způsob doručení pošty a to z důvodů nesplnění nároků kladených právní úpravou, ani bezpečnostních požadavků ve své obvyklé[3] podobě. Daná skutečnost posloužila v rámci České republiky důvodem vzniku tzv. datových schránek, které výše zmíněné problémy řeší (viz zákon č. 300/2008 Sb.[4]). Rysy, které výrazně odlišují datové schránky nad klasickými e-maily jsou: zajišťují identitu odesílatele a platnost dokumentu (dopis se posílá s elektronickým podpisem a časovým razítkem) [12]. Každá odeslaná zpráva je potvrzována doručenkou, která nabízí úplnou datovou informaci včetně výpisu událostí doručení, informaci o okamžiku dodání a doručení datové zprávy. Daná kritéria zajišťují stejnou právní váhu elektronického e-mailu, doručeného do datové schránky, a dokumentu v písemně podobě, který byl poslán do vlastních rukou (zákon č. 300/2008 Sb., § 17 odst. 6). Ve výsledku datové schránky zjednodušují proces doručení dokumentu a řeší problém vyhýbání se dokumentů orgánů veřejné moci. Z výše uvedené informace je patrné, že i z legislativního hlediska e-mailové zprávy jsou čím dál, tím víc právně uznávány a v budoucnu může dojít k nahrazení veškeré komunikace v písemné podobě [12].

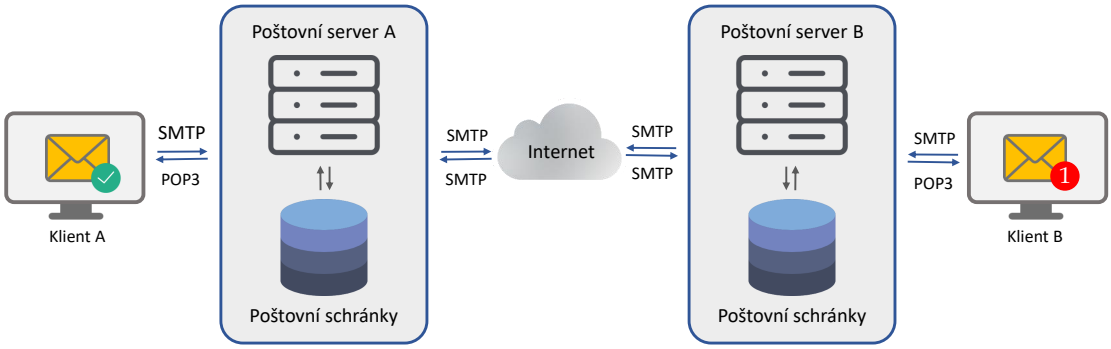
[2]Diagram byl udělán na základě statistik z URL: <`https://www.radicati.com/`>.

[3]V daném případě se jedná o klasické e-mailové zprávy, které nedisponují dodatečnými kryptografickými primitivy na zajištění požadovaných vlastností.

[4]Zákon č. 300/2008 Sb., o elektronických úkonech a autorizované konverzi dokumentů.

1.1 Architektura e-mailové komunikace

Obdobně jako většina primárních služeb na internetu[5], e-mailová komunikace je postavena na modelu typu klient – server [2]. Po tom, až je zpráva napsána a předána do e-mailového systému, putuje mezi mezilehlými servery, dokud se nedostane do e-mailové schránky příjemce. Schematické znázornění procesu přesměrování e-mailu je znázorněno na obrázku číslo [1, 2]. Z obrázku číslo 1.2 je patrné, že se v mailovém systému vyskytují dva typy instancí – poštovní server a klient [2]. Komunikace v tomto systému je založena na principu *store and forward*, to znamená, že pro úspěšné provedení komunikaci není nutné aby komunikující strany byly současně online [1]. Primárním úkolem serveru je provádět směrování zpráv, ukládat poštu, umožňovat přístup jednotlivým klientům apod. Rozsahem působnosti klienta je vytvářet e-mailové zprávy a načítat je z příslušného e-mailového serveru. Úspěšné fungování systému na předání elektronických e-mailů je nejčastěji realizované podporu následujících protokolů aplikační vrstvy: SMTP (*Simple Mail Transfer Protocol*), POP3 (*Post Office Protocol*), IMAP (*Internet Message Access Protocol*). Samotný proces přenosu e-mailů je realizován pomocí protokolu SMTP. Protokoly POP3 a IMAP se používají na straně klienta pro stažení pošty z příslušného serveru [1, 2, 10].



Obr. 1.2: Základní princip e-mailové komunikace

V případě, jestli chce klient *A* (viz obrázek 1.2) kontaktovat klienta *B*, který se nachází v rámci jiné organizace, tak pomocí svého MUA (*Mail User Agent*) agenta[6] zašlé zprávu na příslušný lokální server pomocí SMTP protokolu. Server obdrží zprávu od klienta *A* pomocí svého agenta MTA (*Message Transfer Agent*) a následně rozhoduje kam danou e-mailovou zprávu má přeposlat [1, 2]. V případě, jestli je cílová poštovní schránka lokální, bude doručení elektronické zprávy

[5]Jako podobný příklad je možné uvést HTTP protokol.

[6]V daném případě roli Mail User Agenta (MUA) může zastupovat libovolný e-mailový klient (*mail client*), například Microsoft Outlook, Mozilla Thunderbird atd.

provedeno pomocí MDA (*Mail Delivery Agent*). V opačném případě, tzn. pokud se jedná o vzdáleného uživatele, provede přeposlání e-mailu na další poštovní server [1]. Přenos mezi servery probíhá pomocí protokolu SMTP. Je důležité mít na mysli, že v případě komunikace mezi dvěma servery, odesílací strana zastupuje roli klienta [2]. V praxi to znamená, že server musí disponovat svým vlastním MUA agentem v případě komunikace s jinými e-mailovými servery. Po doručení elektronické zprávy je ji klient *B* schopen stáhnout pomocí protokolů POP3 anebo IMAP [1, 2].

1.2 Formát a struktura e-mailových zpráv

V době svého vzniku formát e-mailových byl poměrně jednoduchý a umožňoval přenos pouze ASCII[7] kódovaných symbolů americké abecedy [1]. S vývojem technologií a rozšířením elektronické komunikace vznikla potřeba přenášet komplexnější data, tzn. obsahující symboly různých znakových sad non-ASCII, multimediální obsah (obrázky, videa, audia), aplikační programy apod. Z takového důvodu byl vytvořen nový standard MIME[8], který rozšiřoval formát klasických e-mailů a umožňoval přenos různých druhů dat. Současně je formát e-mailů spolu s rozšířením MIME definován standardy: RFC 5322, RFC 2045, RFC 2046, RFC 2047, RFC 2048, RFC 2049 [1, 10].

Samotná struktura klasických e-mailových zpráv je poměrně jednoduchá. Podobně, jak i většina datových rámců v komunikačních sítí je struktura e-mailu tvořena dvěma částmi, konkrétně záhlavím (*header*) a tělem zprávy (*body*) [2].

Záhlaví

Syntaxe záhlaví e-mailové zprávy je definována standardem RFC 5322. Táto část e-mailové zprávy je povinná a smí obsahovat pouze tisknutelné znaky. Primárním účelem hlavičky e-mailu je poskytnout informace nutné pro přeposlání a umožnit bezchybné doručení zprávy adresátovi. Hlavička je přesně strukturována, generuje se automaticky při vytvoření a postupně se aktualizuje podle toho jakými servery se prochází. Struktura hlavičky je komplikovaná a může se lišit v závislosti na datech předvyplněných odesílatelem, přenášeném obsahu, nastaveních serveru apod. Jednotlivé položky je možné rozdělit na pole klíčů hlavičky (*header field name*) a pole hodnot hlavičky (*header field body*), které jsou rozděleny dvojtečkou. Důležité je zmínit, že každé pole hlavičky je ukončeno znakem `CRLF`. Vybrané nejdůležitější položky záhlaví jsou uvedeny na další stránce. [1, 2, 10]

[7] ASCII (*American Standard Code for Information Interchange*) – jeden z nejúspěšnějších standardu pro kódování americké znakové sady. Jedná se o sedmibitovou abecedu obsahující 128 znaků.

[8] MIME (*Multipurpose Internet Mail Extensions*) – standard umožnující kódovat a přenášet multimediální a binární data. Primárně byl navržen pro elektronickou poštu, ale současně je implementován i v HTTP protokolu.

- `From` (*Od*)– adresa odesílatele zprávy <`username@example.com`>;
- `Date` (*Datum*) – lokální datum odeslání zprávy nastavený na straně klienta;
- `To` (*Komu*) – adresa příjemce zprávy <`username@example.com`>;
- `Subject` (*Téma zprávy*) – předmět zprávy (vyplněn odesílatelem);
- `Cc` (*Carbon copy*) – adresa příjemce kopie zprávy;
- `Bcc` (*Blind carbon copy*) – adresy použité pouze během SMTP přenosu;
- `Content-type` (*Typ obsahu*) – MIME formát zobrazovaných v mailu dat;
- `Message-ID` – automaticky generována položka identifikující e-mail;
- `In-Reply-To` – položka obsahující identifikaci e-mailu (`Messsage-ID`), v případě, jestli se jedná o odpověď;
- `References` (*Odkazy*) – reference na předchozí e-maily pomocí `Messsage-ID` v případě opakujících se odpovědí;
- `Reply-To` – adresa sloužící k zasílaní odpovědi;
- `Sender` (*Odesílatel*) – adresa uživatele vyřizujícího e-mail, např. sekretářka;
- `Received` – postupně doplněná informacemi z e-mailového serveru přes který zpráva prošla (doména klienta, doména serveru, protokol, časové razítko atd.);
- `Archived-At` – přímý odkaz na archivní verzi e-mailové zprávy.

Výše uvedený seznam není úplný. Záhlaví e-mailové zprávy může navíc obsahovat i další pole, například rodinu tzv. `X`-polí: `X-Country`, `X-Mailer`, `X-Spam-Status`. Jedná se o rozšíření hlavičky[9], která se používají aplikačními programy [10].

Je podstatné mít na vědomí, že pole `To` a `From` nemusí vždycky obsahovat skutečné adresy, které byly použité při doručení zprávy. Skutečný seznam adres je nejčastěji extrahován přímo z tranzitní (*transit-handling*) SMTP obálky, jehož hodnoty můžou, ale nemusí odpovídat hodnotám polí na straně příjemce [3]. Z takového důvodu e-maily bez digitálního podpisu je možné jednoduše zfalšovat a docílit toho, aby koncový uživatel přijal zprávu za důvěryhodnou.

Tělo zprávy

Jak bylo zmíněno dříve původní verze e-mailu umožňovala vkládat do těla zprávy pouze sedmibitové ASCII znaky. Po standardizaci MIME rozšíření bylo možné do těla zpráv vkládat data různých druhů a navíc používat mezinárodní znakovou sadu UTF-8, tzv. *international* e-mail [2, 10].

Přidání MIME rozšíření má zřejmě své výhody, ale na jinou stranu komplikuje strukturu těla e-mailu [5]. V případě použití tohoto rozšíření do hlavičky e-mailu se přidává pole `Content-type`, po kterém obvykle následují surová anebo `base64` kódovaná data. Je důležité zmínit, že tělo e-mailové zprávy může mít vícesložkovou (*multipart*) strukturu [1]. Konkrétně to znamená, že tělo elektronické zprávy

[9]Detaily je možné zjistit z RFC 6648, viz URL: <`https://tools.ietf.org/html/rfc6648`>.

může obsahovat několik různých druhů obsahů (na začátek elektronické zprávy se potom přidá odpovídající pole, např. `multipart/alternative`). V praxi se nejčastěji vyskytují tyto druhy obsahů: `text/plain`, `text/html`, `image/jpeg`, `image/png`, `audio/mp3`, `application/msword` [10].

V případě výskytu v mailové zprávě HTML označkovaného obsahu (`text/html`), z kompatibilních důvodu se do ní, ve většině případů, vloží i kopie posílaného obsahu, avšak bez HTML značek, tj. ve formátu `text/plain` [1].

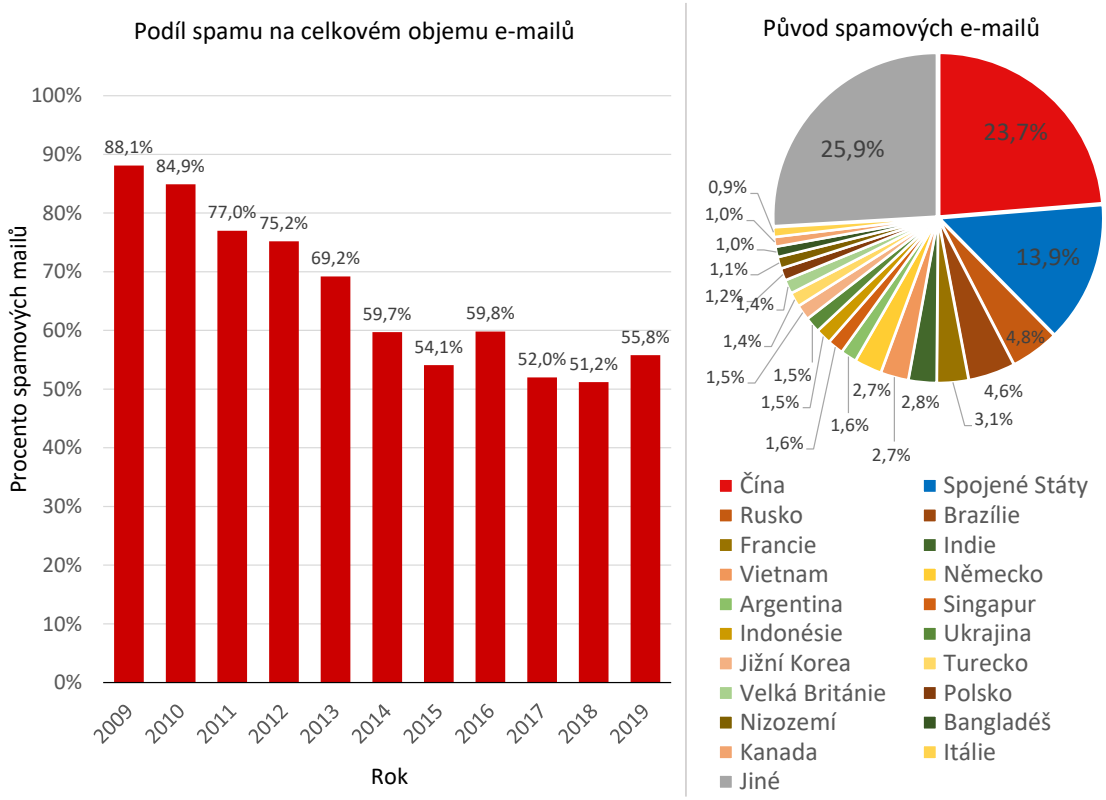
1.3 Základní charakteristika a právní úprava spamu

Obecně se dá říci, že v moderním počítačovém světě je problematika spamu poměrně známa [5, 10]. Od doby vzniku internetových aplikací, které umožnily propojit koncové uživatele z různých částí světa, otázka nevyžádaného obsahu byla čím dál, tím více aktuální. Primární motivaci lidí provádějících hromadné rozesílání nevyžádaného obsahu je snaha o získaní finančního prospěchu, přistup k citlivým údajům oběti, poškození chráněných aktiv anebo pověsti apod. [5] Svět protiopatření, tj. mechanismů umožňujících provádět detekování a odstranění spamu, stejně jak i svět kybernetických zločinců generujících spam, nestojí na místě. Jedná se o nekonečnou válku, zvítězit v které není možné. Denně vznikají nové způsoby obcházení bezpečnostních mechanismu a generování tak sofistikovaného spam obsahu, že ani ty nejmodernější metody detekce nemůžou zabránit jeho šíření a často selhávají. Danou skutečnost je nutné mít na vědomí a provádět aktualizaci a zlepšení existujících mechanismů na detekci, viz podkapitola číslo 1.5.

Samotný pojem *spam* je docela široký, proto je často problematické jej správně interpretovat [5, 10]. V nejširším slova smyslu se jedná o nevyžádané sdělení, které se masově šíří přes internet. Otázkou je, co je možné představit si pod pojmem nevyžádané sdělení. Jelikož každý příjemce může na přijatý „spam“ obsah pohlížet jinak – pro někoho toto sdělení bude nežádoucí a obtěžující, tj. představovat reálný spam, ale pro jiného adresáta, za jistých okolností, představovat užitečnou informaci [10]. Pro opak spamu, tj. pro sdělení, které je cílovým uživatelem považováno za žádoucí a není zasíláno hromadně, se používá termín *ham*, což v doslovném překladu z angličtiny znamená šunka[10]. Je také dobré zmínit, že se v dnešní době pojem *spam* velice často nesprávně chápán a to z hlediska jeho cílového zaměření. Historicky se tento pojem používal vyloženě pro e-mailovou komunikaci a rozesílání nevyžádaných e-mailů [10]. Současně existuje hodně různých druhů projevů spamu. Ve veřejných

[10]Samotná slova *spam* a *ham* nemají nic společného s elektronickou komunikaci, ani s problematikou nevyžádaného obsahu. Jedná se o hovorový pojem, který se historicky asociuje s nevyžádaným obsahem. Primárně se výraz *spam* používal pro kořeněnou šunku a znamenal *Special Processed American Meat.* [14]

blozích je možné se potkat s tzv. *blog spam* [10]. V online chatovacích prostředích s *chat spam* [10]. Další druh spamu – *web-spam* se vyskytuje u online vyhledávačů[11]. Za zmínku také stojí *social spam*, který se objevuje v sociálních sítích, např. Facebook, Instagram, Twitter atd. Další variantou spamu je *sms spam*, během kterého dochází k zasílání nechtěných zpráv na telefonní číslo objeti. [10] Je vidět, že problematika spamu je velmi rozšířena. Tato práce se avšak bude zaměřovat vyloženě na techniky e-mail spamu a jeho variace, nikoliv na obecnou představu o spamu.



Obr. 1.3: Globální statistika výskytu spamových zpráv[12]

Podle provedeného v roce 2019 výzkumu společnosti Kaspersky, každoročně celkový podíl spamových zpráv převyšuje hranici v 50 % (viz diagram číslo 1.3). Na tomto diagramu je vidět statistiku zohledňující podíl spamu na celkovém objemu e-mailů (viz levou část diagramu) a také geografické umístění zemí generujících nejvyšší množství spamových zpráv [6, 7]. Z diagramu číslo 1.3 je patrné, že v roce 2014 došlo k poklesu celkového množství spamových zpráv.

[11]Může se jednat například o tzv. search engine spamming anebo spamdexing, vice je možné dozvědět z URL: <`https://www.webspam.org/seo-spam-what-is-spamdexing/`>.

[12]Diagram byl vytvořen na základě statistických údajů zveřejněných společností Kaspersky v roce 2019, viz [6, 7].

Například oproti roku 2009 se jedná o 28,4% pokles. Jako jeden z důvodů je možné uvést vznik pokročilých metod filtrování spamových zpráv, např. založených na metodách umělé inteligence [10]. V současnosti podíl spamu stále přesahuje hranici v 50 %, v roce 2019 se konkrétně rovnal 55,8 % od celkového množství odeslaných e-mailů. Daná tendence zdůrazňuje skutečnost, že svět kybernetických zločinců nestojí na místě a existence efektivnějších metod filtrace stimuluje spamery hledat nové cesty v obcházení současných metod ochrany. Na grafu vpravo je možné vidět, že největší číslo spamových zpráv bylo v roce 2019 vygenerováno z Číny, konkretně 23,7 %. Druhé místo je obsazeno Spojenými Státy (13,9 %) a třetí – Ruskem (4,8 %) [6, 7].

Nebezpečí skrývající se za spamovými zprávami se nejčastěji kotví v přenášeném obsahu [5, 10]. Nejedná se o pouze reklamní a seznamovací e-maily, které otravují uživatele a zabírají místo v elektronické poštovní schránce. Komplikovaná struktura těla e-mailových zpráv, nedostavené zabezpečení hlavičky a použití rozšíření MIME (viz podkapitola číslo 1.2), umožňují útočníkům modifikovat podobu hlavičky e-mailu, vkládat do těla e-mailové zprávy binární soubory, obrázky, URL odkazy anebo HTML stránky [5]. Ve velké míře je největší nebezpečí zastoupeno nedůvěryhodnými linky vedoucími na podezřelé stránky, přílohami se škodlivým kódem v podobě počítačových červů a vírů apod. Potencionální útočník nejčastěji spoléhá na neznalost a nezkušenost oběti, snaží se zaujmout obsahem e-mailu a znemožnit oběti jednat se s „chladnou“ hlavou [15]. Jako příklad je možné uvést situaci, kdy útočník hromadně rozešle e-mail všem pracovníkům podpory firmy, který obsahuje PDF přílohu s malware jmenující se: „Výplatní páska vedení“[13]. Dalším možným projevem spamu je tzv. *phishing*. Jedná se o druh sociálního inženýrství během kterého dochází k zcizení soukromých údajů uživatelé [15]. Spam je ideální zbraní na šíření malware, proniknuti do chráněné infrastruktury a zcizení aktiv.

Při pohledu na problematiku e-mailového spamu z hlediska právního, je důležité neopomínat, že v případě rozesílání spamových zpráv se nezřídka jedná o činnost mající vztah k vícero státům, potažmo vícero právních řádům (např. odesílatel a příjemce jsou z různých států) [6, 7], což má téměř nezbytně za následek použití norem mezinárodního práva soukromého pro zodpovězení hned několika nesnadných právních otázek. Jedna se zejména o otázku pravomoci správních dozorových orgánů, které budou postupovat vůči správci osobních údajů či v krajním případě ho sankcionovat za porušení norem na ochranu osobních údajů. S pojmem pravomoci (mezinárodní příslušnosti) se nezbytně pojí určení práva rozhodného, podle kterého dozorový orgán bude postupovat a udělovat případné sankce. Nařízení GDPR stanovuje ve svém článku 79 rovněž možnost přímého obracení se na soud pro porušení

[13]Existuje velká pravděpodobnost toho, že tento poutavý název dokumentu, motivuje pracovníka podpory jej stáhnout a otevřít. Jedná se o docela triviální, ale velmi efektivní způsob obejit všechny mechanismy kontroly a dostat vzorek malware do vnitřní infrastruktury firmy [5, 15].

povinností správce (popř. zpracovatele) osobních údajů. S tímto se opět pojí problém určení, soudy kterého státu budou mít pravomoc o takovém nároku rozhodovat a jaké bude použitelné právo pro tento spor. Tyto otázky jdou však nad rozsah a zaměření této práce. V rámci EU, jak již bylo zmíněno, je platné nařízení o ochraně osobních údajů – GDPR (*General Data Protection Regulation*), které se spam problematiky přímo dotýká [13]. Problémem k vyřešení náročnějším však je otázka vymahatelnosti práva EU ve státech nečlenských, na jejichž právní subjekty GDPR v jistých případech svou působnost rozšiřuje. V právním řádu České republiky je účinný zákon č. 480/2004 Sb.[14], který reguluje práva a povinnosti osob poskytující služby informační společnosti a šířící obchodní sdělení – viz § 1 odst. 1. Ustanovení v § 7 odst. 2 zákona č. 480/2004 Sb. definuje, že kontaktování uživatelů elektronickými prostředky je možné využít toliko v případě udělení souhlasu ze strany příjemce, anebo pokud se jedná o zákazníka, jehož elektronický kontakt byl legálně získán v rámci koupi výrobku anebo služby a zároveň byl pořízen v souladu se zákonem o ochraně osobních údajů (zákon č. 110/2019 Sb.[15]). V případě, jestli:

- souhlas nebyl udělen, anebo elektronický kontakt nebyl pořízen legálně
- není možné jednoznačně identifikovat totožnost odesílatele,
- zpráva je zaslána bez platné adresy,
- není možné jednoduše zrušit odběr těchto zpráv,

dojde k porušení právního řadu a vznikne nárok na zahájení řízení na úřadu pro ochranu osobních údajů, který dohlíží na dodržování zákonu č. 480/2004 Sb.

1.4 Technický pohled na evoluci spamu

V době svého vzniku byl kontext spamových zpráv převážné v textové podobě [1]. Do těla e-mailů nebylo možné vkládat žádné sofistikované přílohy, a to z důvodu neexistence MIME rozšíření. Zprávy byly nejčastěji zasílány cíleně na vybrané elektronické adresy a představovaly ze sebe obyčejný surový text (*plaintext*). Z takového důvodu pro zajištění ochrany se používaly bezpečnostní mechanismy umožňující provádět jednoduchou analýzu hlavičky a těla e-mailu [10]. Na základě provedené analýzy bylo rozhodnuto, zda se jedná o spam anebo ham.

S postupem času se situace zhoršila ve prospěch útočníků. Hlavní příčinou tomu posloužila komplikovanější struktura e-mailových zpráv a také popularizace internetu [16]. Útočníci začali používat speciální software na rozesílání spamových zpráv, který jim umožnil zasílat e-maily v shlucích na velké množství cílových adres [10, 16]. Seznamy emailových adres jsou útočníky generovány buď náhodným způsobem anebo pomocí speciálních programů provádějících skenování webových stránek

[14]Zákon č. 480/2004 Sb., o některých službách informační společnosti.
[15]Zákon č. 110/2019 Sb., o zpracování osobních údajů.

a hledání vyskytujících se e-mailových adres [16]. První způsob je nejčastěji technicky vyřešen tak, že útočník použije existující slovníky uživatelských jmen, anebo přizpůsobí je pod konkrétní cílovou skupinu [10]. Přizpůsobení se provádí na základě dodatečných znalosti o cílovém objektu, například se může jednat o situaci, kdy útočník zjistí princip tvorby doménových jmen a upraví hodnoty slovníku tak, aby byla pravděpodobnost doručení spamu co nejvyšší. Druhým způsobem je použití automatických nástrojů na skenování webových stránek [10]. Princip fungování takových algoritmů je založen na *regex* výrazů a předpokladu toho, že většina kontaktních e-mailů je na webových stránkách uložena v otevřeném formátu. Efektivním způsobem ochrany proti těmto nástrojům je provedení tzv. zamaskování (*obfuscation*), například dynamickým generováním kontaktní adresy pomocí JavaScript [16].

Použití jednoduché analýzy obsahu e-mailů (viz podkapitola č. 1.5) nebylo příliš dlouho účinné a z toho důvodu, že útočníci dokázali předpovědět, že budou klasická pro spamové zprávy slova, například „free“, „now“, „discount“ atd. (viz obrázek číslo 1.4), jednoduše detekována na straně příjemce a spamová zpráva nedorazí do poštovné schránky oběti. Pokročilejší techniky spočívaly v použití zamaskovaných (*obfuscated*) slov. Jedná se o techniku během které dochází k skrývaní písmen spamového slova, přičemž se vzhled nevyžádané zprávy při obyčejném[16] prohlížení nemění [10, 16]. Jako příklad je možné uvést použití netisknutelných ASCII symbolů, nahrazení písmen symboly ze vzhledově podobných abeced, maskování slov pomocí HTML značek, použití písmen různé velikosti atd.

Obr. 1.4: Nejpopulárnější anglická slova ve spamových zprávách[17]

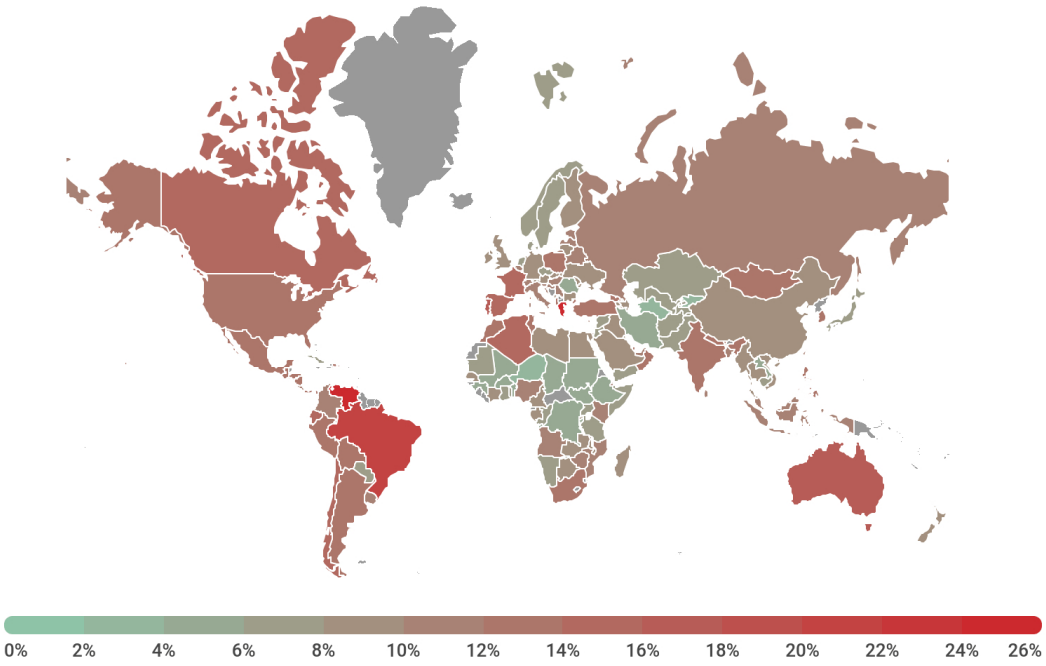
[16] V tomto případě je myšleno prohlížení zpráv pomocí poštovného klienta, nikoliv v plaintextu.

[17] Obrázek je vytvořen na základě statistik z roku 2019 dostupných z URL: <`https://www.activecampaign.com/blog/spam-words`>.

Do seznamu spamových technik je možné počítat *phishing*, viz obrázek číslo 1.5. Jak bylo uvedeno v podkapitole číslo 1.3, jedná se o druh sociálního inženýrství (*social engineering*), který se často uskutečňuje bez přímého kontaktu s obětí, například pomocí nedůvěryhodných elektronických zpráv anebo podvodných webových stánek [15]. V jeho základu leží myšlenka manipulace s lidmi, založená na nedokonalostí jejich rozhodování [15]. Jedná se o velice efektivní a nebezpečný druh útoku, jehož výsledkem může být sdílení citlivých informací, ztratě identity, nakažení malware atd. Pro uskutečnění daného útoku je nezbytné disponovat dobrými znalostmi o cílovém objektu útoku s cílem odhalení jeho slabých míst. Mechanismy útoku při phishingu jsou nejčastěji spamové zprávy, které jsou udělány takovým způsobem, že je velmi problematické rozlišit je od důvěryhodných [5, 10]. Zájmena z důvodu nedostatečné ochrany hlavičky e-mailu (viz kapitola číslo 1.2), útočníci jsou schopni vytvářet e-maily, velice podobné na upozornění z banek anebo bezpečnostních společností, notifikace od poskytovatelů služeb, platebních bran, IT administrátorů firmy, nadřazeného manažera atd. Tato upozornění a notifikace se tváří jako urgentní a nejčastěji vyžaduji přihlášení k internetovým službám, aktualizaci přihlašovacích údajů, převodu peněz anebo sdílení interních dokumentů. Z hlediska textové analýzy se phishingové e-maily můžou tvářit za legitimní, ale obsahovat přitom linky na podezřelé webové stránky, přílohy se škodlivým strojovým kódem [15].

Na obrázku číslo 1.5 je názorně vidět statistiku výskytu phishing zpráv na konci roku 2019. Je možné vidět, že největší množství phishingových zpráv se vyskytuje v Řecku (26,2 %), druhé místo patří Venezuele (25,67 %) a třetí Brazílií (20,86 %). Primární cílovou skupinou útočníků jsou nezkušení uživatelé internetových služeb, zaměstnanci anebo vedení firem, jiné osoby disponující privilegovanými přístupovými oprávněními apod. Účinným způsobem ochrany proti takovým e-mailům je instalace a pravidelná aktualizace anti-spam filtrů a také provedení pravidelného školení uživatelů e-mailových schránek a internetu [5].

K dalším technikám oklamání uživatelů a procházení skrz filtrovací mechanismy je možné zařadit emailové zprávy v jejichž základu leží obrázky se spamovým obsahem (anglicky *image-based spam*) [40]. Jedná se o nevyžádané zprávy, které není možné detekovat použitím klasických textových anti-spam filtrů a to z důvodu absence textové části anebo použití takového textového obsahu, který se jeví jako důvěryhodný [40]. Hlavní technikou vytváření takových e-mailů je vložení spamového plaintext obsahu do obrázků [10]. Tento sofistikovaný druh útoku se výrazně rozšířil v roce 2010 a způsobil obrovský narůst celkového počtu nevyžádaných zpráv (viz obrázek číslo 1.3, období 2009-2013). Způsoby ochrany proti tomuto druhu spamu jsou nejčastěji založeny na OCR (*Optical Character Recognition*) mechanismech, přivádějících analýzu textu v obraze, detailněji viz podkapitola číslo 1.5.



Obr. 1.5: Globální statistika výskytu phishing zpráv[18]

1.5 Způsoby detekce a ochrana proti spamu

Na základě informací uvedených v podkapitole číslo 1.4 a porovnání aktuálních statistik je možné dojít k závěru, že úspěšné filtrování spamových zpráv je velmi problematické, zájmena z důvodu elasticity mechanizmů útoků [10]. Daná skutečnost je jakýmsi podnětem k provedení komplexního skenování obsahu e-mailových zpráv v kombinaci s dalšími mechanismy ochrany. Před tím, než budou uvedeny technické způsoby detekce spamových zpráv, je také dobré zmínit, že technické prostředky zajištění bezpečnosti jsou pouze základem, který musí být rozhodně doplněn znalostmi a zodpovědným přístupem koncových uživatelů [5].

Pokud se detailněji seznámíme se schematem číslo 1.2, je možné dospět k závěru, že detekční mechanizmy je možné technicky implementovat buď na straně koncového uživatele, anebo na straně poštovního SMTP serveru [16]. V dnešní době je také možné se potkat s kombinací těchto dvou způsobů [10]. Tendence vývoje cloudových služeb a centralizace výpočetního výkonu posloužily důvodem, proč se filtrování nevyžádaných e-mailů nejčastěji řeší na straně poštovního serveru [10]. Obecně se dá říci, že daný způsob zajištění bezpečnosti je účinnější, poněvadž server disponuje vyšším výpočetním výkonem, má větší přehled nad obsahem poštovních

[18]Diagram je dostupný ze statistik zveřejněných v roce 2019, viz následující zdroje [6, 7].

schránek uživatelů a zároveň je schopen odmítnou příjem nové pošty již v rámci navázání SMTP relace. Na druhou stranu do nevýhod takového způsobu je možné zařadit nedostatečnou flexibilitu řešení a závislost na bezpečnostních mechanismech na straně serveru [16]. V případě realizace filtrování na straně koncových stanic je možné docílit větší flexibility a nezávislosti na bezpečnostních řešeních serverů s výhradou na přenesení výpočetní zátěže na koncová výpočetní zařízení.

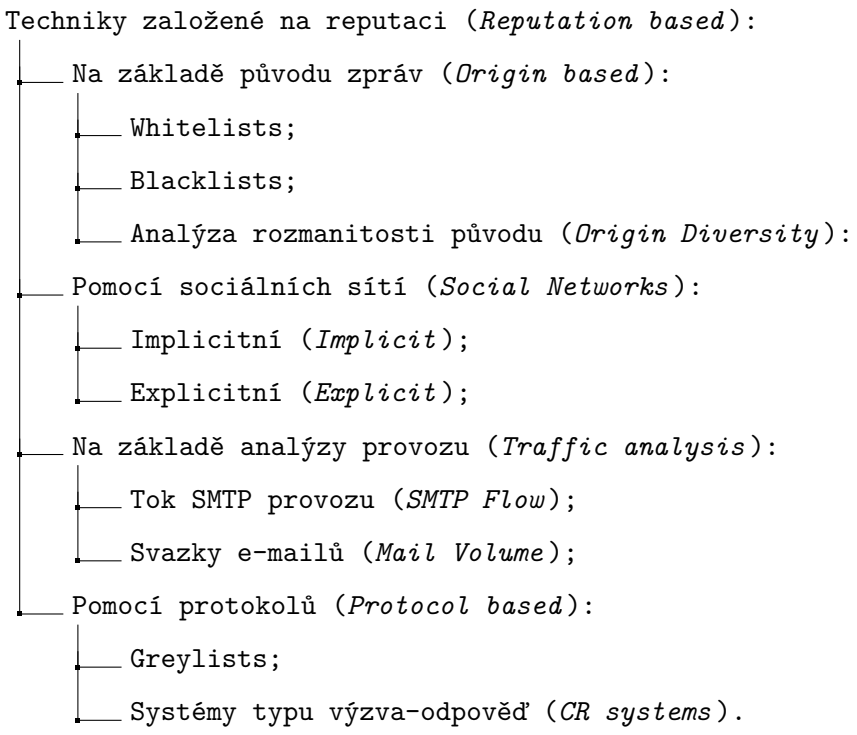
Jedním z možných způsobů rozdělení mechanismů detekování nevyžádané pošty je rozdělení podle reputace zprávy a analyzovaného obsahu [10]. Současné techniky mohou být rozdělené do třech hlavních skupin:

1. Založené na reputaci (*Reputation-based*);
2. Založené na analýze kontextu (*Textual content-based*);
3. Založené na analýze multimediálního obsahu (*Multimedia content-based*).

Analýza na základě reputace (*Reputation-based analysis*)

Do této skupiny (viz obrázek číslo 1.6) je možné zařadit techniky, jejichž primárním účelem není provádět analýzu kontextů a obsahu e-mailových zpráv, ale provádět řízení e-mailové komunikace v době její zahájení, například přímo na MTA agentu. V této skupině se objevují techniky založeny jak na analýze SMTP provozu (*SMTP Flow*) a svazků e-mailů (*Mail Volume*), tak i pokročilé, které přivádí analýzu sociální sítě uživatele (*Social Network Based*) a generují seznamy důvěryhodnosti odesílatelů.

Výhoda všech uvedených způsobů spočívá v tom, že je možné přenos spamových zpráv odmítnout již během navázání SMTP spojení [10, 16]. Dané techniky se považují za docela efektivní a odolné vůči distribuovaným shlukům nevyžádaných zpráv. Do nevýhod lze zařadit větší režii, závislost na externích zdrojích informací a externích službách, které mohou tvořit slabé místo, tj. *bottleneck* filtru. V případě použití externích zdrojů dat, problém se může skrývat v nedůvěryhodných anebo neaktuálních informacích poskytnutých třetí stranou. Tyto informace mohou být podvržené, obdržené z nelegitimních zdrojů, nedostupné anebo zastaralé. Tyto faktory ve finále negativně ovlivní proces klasifikace a tím zvýší pravděpodobnost nedetekování reálného spamu. V případě vedení lokálních reputačních seznamů, je nutné zajistit jejich pravidelnou aktualizaci a logování s cílem zajištění účinného filtrování a nenarušitelnosti legitimního provozu. Pokud se v rámci *reputation based* technik zaměříme na spolupráci s externími služby, je dobré dbát na snížení latenci, prioritizaci daného provozu a nadefinování záložních zdrojů informací [16]. Populárními představiteli technik založených na reputaci jsou filtry provádějící svoji činnost na základě původu zpráv, neboli *origin based* filtry. Primárními představiteli dané skupiny jsou klasifikátory postaveny na *blacklist* a *whitelist* seznamech anebo na provedení analýzy rozmanitosti původu zprav, tj. *origin diversity analysis*.



Obr. 1.6: Rozdělení technik detekce založených na reputaci

Princip fungování *blacklist* filtrů spočívá v tom, že tato technika používá distribuovanou databázi obsahující IP adresy a DNS záznamy nedůvěryhodných odesílatelů, z jejichž adres byl v minulosti distribuován nevyžádaný obsah [10]. Hledání potencionálně nebezpečných IP adres a DNS záznamů se také může provádět v položkách hlavičky zprávy, anebo v samotném těle, například při výskytu v textu podezřelých URL odkazů. *Whitelist* filtry, neboli filtry založené na důvěryhodných seznamech, jsou opakem *blacklist* filtrů [5]. V jejich jádru leží myšlenka povolení SMTP komunikace pouze z předem určených adres uvedených ve seznamu. V případě, jestli na poštovní server dorazí elektronická zpráva z adresy, která není evidována ve *whitelist* seznamu, bude taková zpráva klasifikována za nedůvěryhodnou a následně buď zamítnutá, anebo označkována a propuštěna dál[19]. *Whitelist* seznamy nejčastěji nejsou distribuované, tzn. jsou vedeny lokálně [16]. Velikost seznamu je relativně malá a v případě jejich použití je eliminované riziko nedostupnosti externího zdroje dat. Do nevýhod daného přístupu je možné zařadit uzavřenost infrastruktury a problematická komunikace s účastníky, jejichž adresy nejsou ve *whitelist* seznamech vedeny. Poslední metoda filtrace v této kategorie je analýza rozmanitosti původu, neboli *origin diversity analysis*, která spočívá v hledání statistických závislostí mezi

[19]Způsob zacházení SMTP serveru s takovou elektronickou zprávou se může lišit v závislosti na provedených konfiguracích a preferencích správce.

množinou odesílatelů a množstvím jimi vygenerovaných zpráv. Funkčnost dané techniky odráží skutečnost, že jsou nevyžádané zprávy nejčastěji rozesílány ve shlucích, které tvoří anomálie v klasickém SMTP provozu, detailněji viz [10].

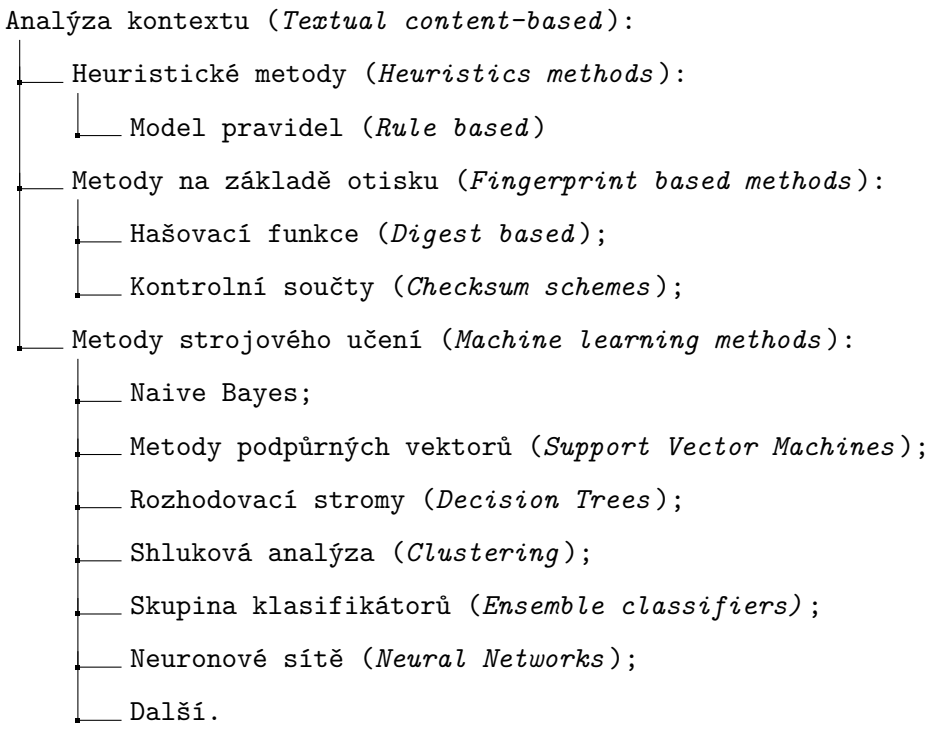
Systémy založené na analýze sociální sítě uživatele (*Social Network based*) provádí analýzu hlaviček odesílaných a přijímaných zpráv a vytváří graf sociální propojenosti uživatele [10]. Tyto techniky je možné rozdělit na implicitní (*implicit*) a explicitní (*explicit*). Samotná klasifikace e-mailů probíhá na základě tzv. clustrovacího koeficientu (*clustering coefficient*), který je vypočítán pro jednotlivé uzly grafu, tj. uživatele. Daný koeficient je velmi nízký pro nelegitimního uživatele, protože nemá tak těsné sociální vazby s ostatními členy. Naopak pro jednu firemní divizi, nebo skupinu přátel bude hodnota tohoto koeficientu vysoká, protože uzly v subgrafu budou hustě propojeny (*dense subgraph*).

Další významnou technikou v této skupině je filtrace na základě *greylists* seznamů. V její základu leží použití vlastnosti SMTP protokolu, konkrétně dočasného odmítnutí zpráv (SMTP zprávy s kódem 450-459). Po provedení dočasného zamítnutí legitimní server nejčastěji provede pokus o opakované odeslání a tím bude přidán do whitelistu s následující akceptaci přijaté zprávy [16]. V případě nevyžádané pošty, zprávy se rozesílají ve shlucích a po dočasném zamítnutí se většinou nepřeposílají. Danou techniku je možné zkombinovat s lokálními *Blacklists*, což výrazně zefektivní účinnost filtrování na základě reputace [10]. Za zmínku také stojí systémy typu výzva-odpověď, neboli *challenge response systems*, které po přijetí zprávy zasílají výzvu s cílem ověření legitimnosti odesílatele a jeho prezenci na protější straně. Autentizace se provádí buď na základě jednoduché otázky, anebo použitím složitějších nástrojů, např. CAPTCHA (*Completely Automated Public Turing test to tell Computers and Humans Apart*), detailněji viz [10].

Analýza textového obsahu (*Textual content-based analysis*)

Princip fungování filtrů uvedených na obrázku číslo 1.7 je koncipován na provedení analýzy textového obsahu přijatých zpráv [10]. Analýza se může provádět buď na základě statistického vyhodnocení obsahu přijaté zprávy (*Heuristics methods*), otisku zprávy (*Fingeprint based methods*) anebo použitím sofistikovanějších metod založených na strojovém učení (*Machine learning methods*). Tyto tři metody filtrování jsou považovány za velice účinné a po provedení analýzy je zpráva s velkou pravděpodobností správně zařazena do příslušné kategorie, tj. spam anebo ham [10].

Do první skupiny metod provádějících analýzu kontextu spadají tzv. heuristické metody (*Heuristics methods*). Tyto metody staví na principu znalostního inženýrství (*knowledge engineering*) a provádí klasifikaci přijatých e-mailů na základě specifických příznaků, charakterních pouze spamovým e-mailům [10, 11].



Obr. 1.7: Rozdělení technik detekce založených na analýze kontextu

Jedná se například o techniku filtrování založenou na pravidlech (*Rule based*), která se zaměřuje na hledání výskytů rozšířených „spamových“ slov například: „*free*“, „*money*“, „*viagra*“ (viz obrázek 1.4) anebo specifických strukturálních vzorů v e-mailech. Takové metody ve svém základu nejčastěji obsahují seznam klíčových slov a regulárních výrazů (*regex patterns*), které na základě nalezených shod přiřazují ověřované zprávě, tzv. váhu důvěryhodnosti. Do silné stránky takových metod je možné zařadit jejich jednoduchost, regulovatelnost a účinnost. Do nevýhod *rule based* modelu spadá nutnost pravidelné aktualizace seznamu klíčových slov a pravidel, provádění dodatečného ladění při změně nastavení a také nízkou odolnost vůči sofistikovaným útočníkům provádějícím zamaskování slov elektronické zprávy pomocí technik *word obfuscation*, detailněji viz podkapitola číslo 1.4.

Do druhé skupiny spadají metody provádějící klasifikaci e-mailů na základě jejich unikátního otisku (*Fingeprint based methods*). Podobné techniky se také úspěšně aplikují v antivirových aplikacích realizujících klasifikaci binárních souborů na základě signatur [5]. Princip fungování *fingeprint based* metod je celkem jednoduchý a spočívá ve vytvoření individuální charakteristiky ověřovaným datům. Jedná z možností vytvoření takové charakteristiky je využití speciální hašovací funkce (*hash function*), která je schopná na základě jakkoliv dlouhého vstupu vygenerovat řetězec pevné délky, který jednoznačně identifikuje vstupní data, tzv. vystupuje unikátním

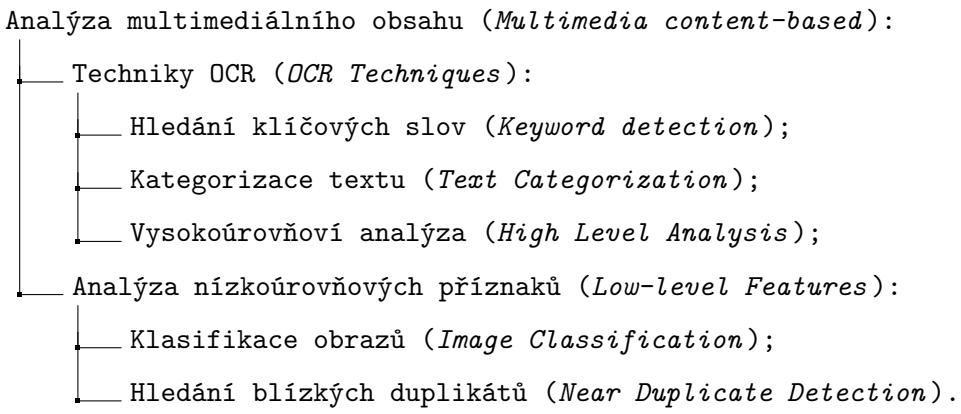
otiskem zprávy [5]. Pro tyto *digest based* funkce platí: jsou jednocestné (není možné zpětně získat vstupní data), výstup má pevnou délku, velká úroveň difúze (rozprostření statistických vlastností vstupních dat do výstupného řetězce), bezkoliznost statisticky omezená délkou haše (problematické hledání různých vstupů generujících stejný výstup) [5]. Po vygenerování otisku přijaté zprávy, SMTP server ověřuje zda podobný otisku již nevyskytuje v jeho lokálním seznamu, případně v distribuované databázi otisku spamových e-mailů [10]. V případě nalezení shody je zpráva automaticky zařazená do spamové složky. Do nevýhod daného způsobu detekce je možné zařadit nutnost udržování a pravidelné aktualizace databáze otisků, slabá odolnost vůči změně kontextu, kvůli velké úrovni difúze.

Poslední a v dnešní době nejpřesnější metoda klasifikace e-mailů je založena na technikách strojového učení (*Machine Learning*). Do těchto technik je možné zařadit: Naive Bayes – jednu z nejpoužívanějších technik na analýzu spamu, Metody podpůrných vektorů (*Support Vector Machines*), Rozhodovací stromy (*Decision Trees*), Shlukovou analýzu (*Clustering*), Skupiny klasifikátorů (*Ensemble classifiers*) a Neuronové sítě (*Neural Networks*) – moderní a velice slibná technika spam detekce [10]. V základě výše zmíněných technik leží myšlenka vytvoření flexibilního systému, nejčastěji v podobě binárního klasifikátoru, tj. nevyžádaná pošta (−) anebo e-mailová komunikace legitimního uživatele (+), který by inteligentně dokázal definovat statistické závislosti mezi vstupními a výstupnými daty, automaticky zlepšovat se a vyvíjet [10]. Pro vytvoření modelu, založeného na mechanismech strojového učení, je důležité disponovat překlasifikovanou datovou sadou[20], které se říká *training set* anebo *initial corpus* [10]. Čím kvalitnější a co nejvíce pokrývající vybraná datová sada bude, tím je pravděpodobnost vytvoření robustního a přesného modelu vyšší [17]. Pokud provedeme porovnání způsobu detekce založeného na strojovém učení a *rule based* modelu, tak dospějeme k závěru, že v případě *rule based* modelu je nutné neustále udržovat a aktualizovat databázi pravidel, což ve výsledku může být velmi časově náročné a pracné [10, 16]. V případě modelu založeného na strojovém učení výhoda tohoto způsobu je zřejmá – systém není postaven na statických pravidlech, ale je dynamický, tj. je schopen provádět klasifikaci takových dat, která při jeho učení nevyskytovala. Ve výsledku použití této technologie vyžaduje menší investici času na údržbu, při zachování vysoké úrovně přesnosti [10]. Podrobný popis výše uvedených technik bude proveden v kapitole číslo 2.5 a také ve zdrojích [10, 11].

[20] Avšak existují problémy, v nichž není příprava klasifikované datové sady možná. V takových případech se aplikuji jiné techniky strojového učení, viz kapitola číslo 2.3.

Analýza multimediálního obsahu (*Multimedia content-based analysis*)

Do třetí skupiny technik je možné zařadit ty, jejichž fungování je založeno na principech analýzy multimediálních dat (viz obrázek číslo 1.8). Jednou z příčin vzniku této skupiny posloužilo přidání rozšíření MIME do struktury e-mailů a možnost vytvoření multimediálního spamu [16]. Daná skutečnost způsobila strmý nárůst spamových zprav v rocích 2009 až 2012 (viz diagram číslo 1.3). Základními technikami filtrování takového spamu jsou techniky OCR, neboli *optical character recognition* a techniky hledání *Low-level Features* [10, 11].



Obr. 1.8: Techniky detekce založené na analýze multimediálního obsahu

V bezpečnostním světě se techniky OCR objevily jako první v boji proti nevyžádanému multimediálnímu spamu. V jejich základu ležela myšlenka extrahování textu z analyzovaných elektronických zprav. V dalších krocích byl rozpoznaný text porovnán s distribuovanou databází spamových signatur [10], anebo podroben klasické analýze kontextu, viz podkapitola číslo 1.5. Do seznamu výhod výše uvedeného způsobu je možné zařadit možnost aplikování textové analýzy na multimediální data bez nutnosti zapojení dodatečných technik klasifikace [10]. Do nevýhod OCR patří rychlost, závislost na externím zdroji dat a vysoká výpočetní náročnost při provedení primární extrakce symbolů [10]. Do seznamu těchto technik je možné zařadit hledání klíčových slov (*keyword detection*), kategorizaci textu (*text categorization*) a vysokoúrovňovou analýzu příznaků (*high level analysis*).

Druhou skupinu mechanismu na provedení analýzy multimediálního obsahu je Analýza nízkoúrovňových příznaků (*low-level features*). Tyto systémy jsou postaveny na extrahování určitých příznaků a závislostí při analýze obrazů. Oproti OCR metodám jsou efektivnější a to zájmena z důvodu nižších výpočetních nároků při provozu a také odolnosti vůči maskování (*obfuscation*) a znáhodnění obsahu (*randomization*). Takový přístup umožňuje dosáhnout nižší míry *false-positives*, detailněji o těchto technikách je možné dočíst z [40].

2 Filtrování e-mailových zpráv pomocí metod umělé inteligence

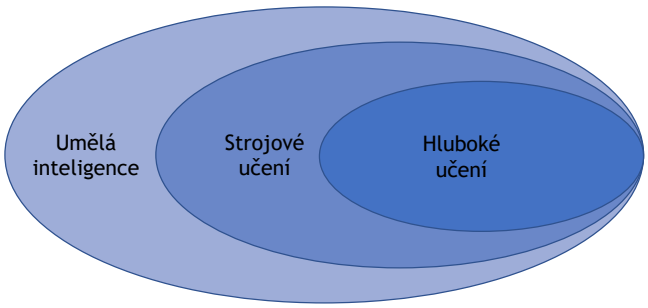
Technologický pokrok nestojí na místě. Denně vznikají technologické vynálezy a inovace, které směrují vývoj lidského druhu *homo sapiens* exponenciálně dopředu. V minulosti takovými technickými objevy byly parní stroj, vynález elektřiny, vznik počítače a internetu. Tyto objevy kardinálně změnily existující procesy ve společnosti a posunuly ji na novou úroveň evoluční spirály [18, 28]. Ti jedinci, kteří dané technické objevy nevyužily, minuli se s tou dobou, která nevratně nastala. Jedním z takových technologických objevu je možné považovat vznik umělé inteligence a stojící za tím vývoj výpočetní techniky, který umožnil paralelizovat a mnohonásobně zrychlit proces učení počítačových systémů [18]. V roce 1967 známý americky vědec Martin Lee Minsky, který se zabýval technikami umělé inteligence dokázal velmi přesně definovat tento pojem: „*Umělá inteligence je věda o vytváření strojů nebo systémů, které budou při řešení určitého úkolu užívat takového postupu, který – kdyby ho dělal člověk – bychom považovali za projev jeho inteligence*“ [29].

Samo o sobě, odvětví umělé inteligence, kromě zmiňovaných v textu oblastí strojového učení, zahrnuje v sobě mnoho dalších přístupů, které lze aplikovat na různá technická odvětví a tím zautomatizovat a zefektivnit jejich procesy [18]. Jedním z takových technických odvětví je počítačová bezpečnost, která přístupy umělé inteligence aktivně využívá, například s účelem rozpoznání nedůvěryhodného obsahu a mitigaci vektorů útoků uskutečňovaných přes e-mailovou komunikaci.

V rámci této kapitoly bude udělán teoretický rozbor problematiky strojového učení a způsobů realizace filtrování spamových zpráv pomocí hlubokých neuronových sítí. V textu kapitoly budou vysvětleny nejdůležitější principy a pojmy, které je možné ve světě strojového učení potkat. Pochopení výše zmíněných základů je nezbytné, a to jak při studiu technické literatury, tak i při provedení učení a ladění neuronových sítí. Na začátku dané kapitoly důraz bude kladen na vysvětlení základních vlastnosti neuronových sítí a principů jejich fungování. V rámci druhé podkapitoly dojde k popisu a srovnání existujících způsobů učení neuronových sítí. Následně, pozornost bude věnována problematice rozpoznání přirozeného jazyka (*natural language processing*) a klasifikaci textového obsahu. V podkapitole číslo 2.4 budou popsány moderní architektury neuronových sítí a způsoby realizace přeneseného učení (*transfer learning*), které se v dnešní době jeví jako nejvíce úspěšné. Popsáné v rámci této části principy budou přímo aplikovány na provedení textové klasifikace spamových zpráv, viz kapitola číslo 3.4. Na konci kapitoly budou stručně porovnány existující v současné době *state-of-the-art* techniky, které je možné aplikovat na problém detekování spamových zpráv.

2.1 Úvod do problematiky hlubokého učení

Hlavní důraz je v rámci této práce kladen na vyřešení problému filtrování e-mailových zpráv pomocí metod umělé inteligence, aplikováním technik strojového učení. Jak bylo zmíněno výše, je obor umělé inteligence velice rozsáhlý a zahrnuje v sobě mnoho různých přístupů, například genetické algoritmy (*genetic algorithms*), prohledání stavového prostoru (*state space search*), expertní systémy (*expert systems*) atd. Některé z nich proces učení nezahrnuji a mohou být postaveny na souborech explicitních pravidel ručně vymyšlených programátory [18]. Avšak jedním odvětvím, které myšlenku „učení se“ aktivně využívá je strojové učení. Toto odvětví vychází z myšlenky vytvořit takový počítačový systém, který by byl schopen překonat hranice toho, co mu programátor může nařídit a byl schopen samostatně se přizpůsobit měnícím se podmínkám okolí [18]. V doslovném slova smyslu takové systémy jsou natrénovány, než ručně naprogramovány. V důsledku trénování dochází k dolování specifických vlastností ze vstupních dat a vytvoření statistických závislosti mezi nimi [28]. Na základě vytvořených závislosti, které nemusí být pro obyčejného člověka zcela zřejmé, je systém schopen provádět klasifikaci a automatizaci požadovaných řešení [18]. Jednou z moderních podskupin strojového učení je hluboké učení, které je založeno v použití několika jdoucích za sebou vrstev reprezentace znalostí. Na obrázku číslo 2.1 je možné vidět hierarchické zařazení odvětví hlubokého učení do oboru umělé inteligence a strojového učení [18]. Hloubka systému založeného na principech hlubokého učení závisí od počtu vrstev reprezentace – v moderních systémech se může jednat o desítky anebo stovky jdoucích za sebou vnitřních vrstev. Silnou stranou hlubokého učení je mnohonásobně vyšší schopnost abstrakce vstupních dat a odhalení v nich skrytých závislostí. Problematickou části je naučení daného modelu, jelikož vyšší počet vnitřních vrstev znamená vyšší finální složitost systému a s ní plynoucí vyšší počet operací nutných na realizaci procesu trénování. Zároveň, kromě vyšší asymptotické složitosti, hluboké sítě mají i vyšší prostorové (paměťové) nároky.

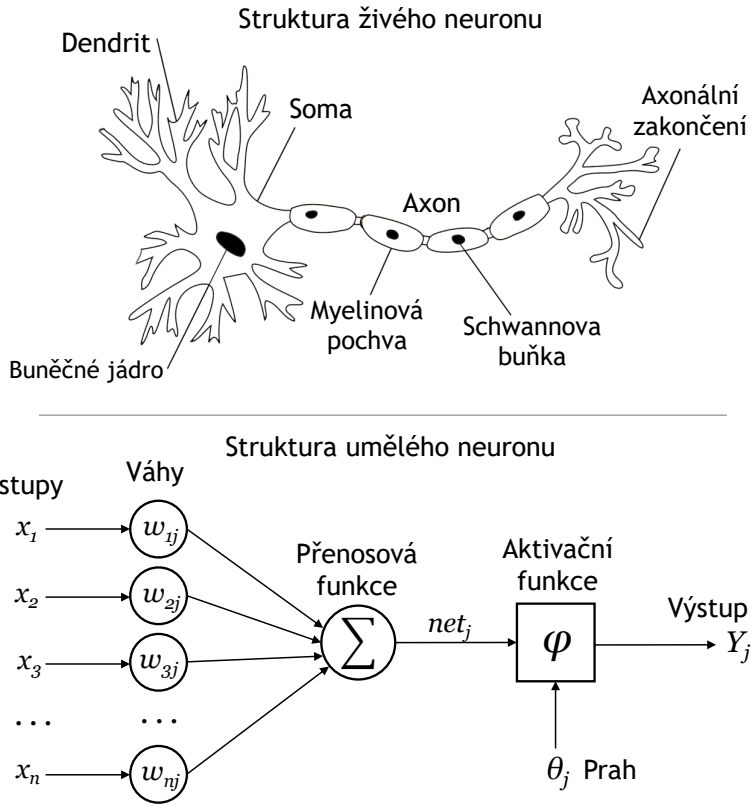


Obr. 2.1: Umístění hlubokého učení v oboru umělé inteligence

2.2 Základní vlastnosti neuronových sítí

Pokud se zaměříme na problematiku umělých neuronových sítí, hned zjistíme, že se jedná skupinu učících se výpočetních modelů, které se používají v oboru strojového učení [18]. Termín neuronová síť je přímým odkazem na vědu o principech fungování nervové soustavy živočichů – neurologii, která je součástí neurovědy. Znamená to, že v samotném základu umělých neuronových sítí leží neurofyziologické poznatky o principech fungování lidského mozku [18]. Dá se říci, že v případě umělých neuronových sítí se jedná o velmi zjednodušený, matematický pohled na modely nervových systémů živočichů, který prokázal v praxi svoji funkčnost. Po provedené simulace takových umělých systémů bylo zjištěno, že natrénované sítě jsou schopny vykazovat vlastnosti velmi blízké k přirozené lidské inteligence [28]. Konkrétně se jednalo o vlastnost učit se na základě vstupních informací, zobecňovat naučenou informaci a vytvářet mezi ní jednoduché souvislosti. Avšak je důležité mít na mysli, že se stálé pohybujeme v odvětví teoretické informatiky, matematiky a statistiky, nikoliv v biologii. Stoprocentně není možné tvrdit, že modely umělých neuronových systémů jsou kopiemi mozku, protože neexistuje žádný potvrzený důkaz o tom, aby lidský mozek fungoval na podobných matematických mechanismech jako dnešní umělé neuronové sítě [18]. Navíc není možné z teoretického hlediska jednoznačně říci, co znamená tvrzení: „Neuronová síť se chová inteligentně“ [28]. Správním je tedy pohled pouze konceptuální, nikoliv funkcionální blízkosti mozku a umělého neuronového systému. Zároveň není možné pohlížet na neuronové sítě jako všemocný nástroj datové reprezentace. Ačkoliv se v praxi se ukázalo, že jsou neuronové sítě velmi efektivním nástrojem na provedení klasifikaci, aproximaci a predikci nad vstupními daty, stálé mají své nevýhody v podobě časové složitosti nutné na provedení učení, nepředvidatelnosti finálního výsledku a těsné propojenosti s kvalitou vstupních dat [17, 18, 28].

Princip fungování umělých neuronových sítí se dá popsat jako vícestupňový proces filtrování a extrahování klíčových příznaků ze vstupních dat [19]. V rámci daného procesu dochází k mapování vstupů na výstupní (cílová) data. Samotné mapování je docílené postupnou transformací informace pomocí vnitřních vrstev. Jak bylo zmíněno výše, konceptuálně jsou neuronové sítě velmi blízké ke svým biologickým modelům, v jejichž základu leží biologické neurony [28]. Základním stavebním prvkem umělých síti je perceptron (matematický model biologického neuronu). Primárním cílem takového neuronu je zpracovávat vstupní impulzy, které můžou nadcházet buď z vnějšího okolí anebo z bliž ležících neuronů [18]. Proces zpracování může spočívat například v předání vstupného impulzu na svůj výstup a tím docílení šíření vstupného signálu. Následně neuron může provádět zpracování impulzu a také jeho dočasně ukládat. Na obrázku číslo 2.2 je schematicky uveden model zjednodušeného umělého neuronu (perceptronu) v porovnání s živým neuronem.



Obr. 2.2: Srovnání struktur živého a umělého neuronu

Pokud se zaměříme na organický neuron – je to nervová buňka, která je základem pro vybudování nervových tkání. Primárním účelem biologického neuronu je převod informací a generování odpovědí na budicí signály [28]. Neuron se skládá z těla neuronu a výběžků (krátkých - dendritu a dlouhého - axonu). Tělo neuronu, tzv. soma složí jako energetický centrum a obsahuje ve svém základu mitochondrie. Výběžky jsou ukončeny axonálním zakončením, sloužícím k napojení na blíž ležící neurony. Tyto výběžky berou účast v realizaci synapse [28].

V případě umělých neuronových sítí se jedná o jakési matematické formulování a napodobení funkcí živého neuronu [28]. V závislosti na typu a charakteru úlohy je samotný neuron nejčastěji definován množinou reálných n vstupů, označených: $x_1, \ldots, x_i$. Každý z těchto vstupů napodobuje dendrity živého neuronu a má přiřazenou vlastní váhu w_i, tzv. synaptickou váhu. Tyto váhy určují propustnost vstupních kanálů a můžou nabývat jak kladných tak i záporných hodnot [18, 28]. V případě, jestli je synaptická váha záporná, znamená to, že se jedná o inhibiční (způsobující útlum) charakter vstupního signálu [28]. Všechny vstupy x_i a do těla neuronů jsou sloučeny zváženou sumou ξ v rámci přenosové funkce:

$$\xi = \sum_{i=1}^{n} x_i w_i \,.$$

Tato zvážená suma zastupuje vnitřní potenciál neuronu, který po dosažení definované prahové hodnoty symbolizuje impulz neuronu. Generování výstupního impulzu na základě prahové hodnoty je v případě umělých neuronů popsáno aktivační funkcí ϕ [28]. Existuje hodně různých druhů aktivačních funkcí [18, 28], například: funkce jednotkového skoku, aktivační funkce signum, lineární aktivační funkce, sigmoidální aktivační funkce atd. Finální výstupní signál neuronu Y je tedy lze matematicky definovat následujícím způsobem:

$$Y = \phi\left(\xi + \theta\right) = \phi\left(\sum_{i=1}^{n}(x_i w_i) + \theta\right).$$

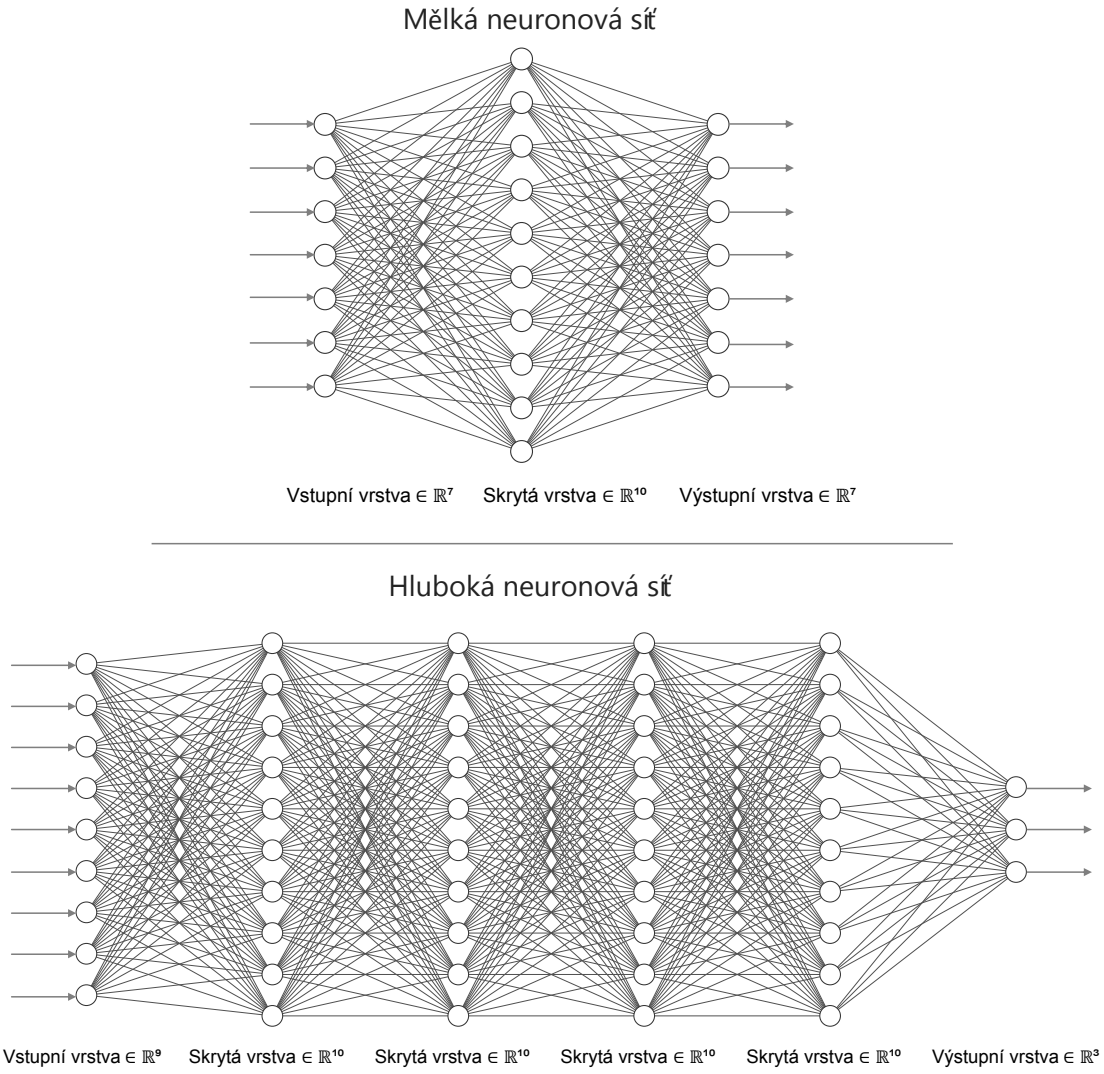
Je důležité rozumět, že sám o sobě jednoduchý umělý neuron, nemůže prokazovat inteligentní vlastnosti, jelikož má poměrně jednoduchou matematickou strukturu [28]. V praxi jsou tyto stavební prvky nejčastěji spojeny do vrstev, kde každá plní určitou funkci a funguje na principu filtru, který umožňuje vstupní informace generalizovat a následně mapovat na množinu výstupních charakteristik [19]. Obecně se dá říci, že výstup jednoho umělého neuronu je vstupem mnoha dalších neuronů ležících na hlubší vrstvě. Na neuronovou síť je možné tedy pohlížet jak na grafovou strukturu, jejíž architekturu tvoří vzájemně propojené vrstvy (můžou být i plně propojené). Z funkčního hlediska je možné v rámci neuronové sítě definovat tři druhy vrstev:

1. Vstupní vrstva (*Input layer*) – primárním účelem vstupní vrstvy neuronové sítě je přijímat povzbuzující signál z vnějšího okolí a transformovat jej do takové podoby vhodné na zpracování vnitřními vrstvami [28];
2. Skryté vrstvy (*Hidden layers*) – slouží k přenosu a transformaci povzbuzujících signálů – tvoří jádro a největší generalizační sílu modelu. Existence vnitřních vrstev umožňuje neuronovým sítím projevovat inteligentní schopnosti [28];
3. Výstupní vrstva (*Output layer*) – primárním účelem těchto vrstev je převod signálu ze skrytých vrstev do vnějšího okolí. V případě pokud se jedná o klasifikační problém, výstupní vrstva obvykle tvoří finální klasifikační třídy, které budou přiřazeny vstupním datům [28].

Počet skrytých vrstev v rámci neuronové sítě závisí na použité architektuře a definuje hloubku umělého modelu (viz obrázek číslo 2.3). V praxi můžeme rozlišit hluboké neuronové sítě DNN (*Deep Neural Network*) a mělké neuronové sítě SNN (*Shallow Neural Network*) [18]. V případě hlubokých neuronových sítí počet vnitřních vrstev se může pohybovat od desítek do stovek, co ve výsledku umožňuje dosáhnout velké úrovně abstrakce nad vstupními daty a řešit komplexní problémy [18]. Problém, který přichází s velkým množstvím vrstev spočívá v problematickém učení takových modelu, z hlediska jejich asymptotické a prostorové složitosti, detailněji viz podkapitola číslo 2.1. V případě mělkých neuronových sítí se jedna o takové modely, jejichž hloubka je oproti DNN architekturám malá (nejčastěji jedná vnitřní vrstva).

Takové sítě jsou relativně jednoduché na trénování, jelikož neobsahují velké množství vnitřních vrstev a jsou vhodné na aplikování jednodušších druhů problémů [19].

Pro lepší pochopení principů fungování neuronových sítí je důležité zmínit, že procesy transformace vstupních dat, které na vnitřních vrstvách probíhají, přímo závisí na vahách uložených v rámci jednotlivých neuronů [19]. Samotný proces učení sítí spočívá v přizpůsobení množiny vah každé z vrstev, a to taký způsobem, aby výsledná hodnota ztrátové `Loss` funkce (*loss score*) byla co nejmenší [18, 19]. Samotné ztrátové skóre (*loss score*) odráží, to jak silně se trénovaná síť spletla oproti očekávanému na výstupu výsledku. U nějakých druhů metod strojového učení se finální *loss score* používá pro provedení zpětného nastavení vah [19]. V takovém případě dochází k implementaci techniky *backpropagation learning* [18, 19].



Obr. 2.3: Srovnání architektury hlubokých a mělkých neuronových sítí

2.3 Způsoby učení neuronových sítí

Svět strojového učení je velmi široká oblast, ve které existují různé metody na vyřešení specifických typů problémů [17, 18]. V dnešní době je možné se potkat s takovými problémy jako binární klasifikace, klasifikace do několika tříd, skalární regrese apod. Takové problémy se nejčastěji řeší technikou učení s učitelem, tj. přístupem, ve kterém se vyskytuje prvek řídící celý proces [18]. Avšak kromě učení s učitelem je možné definovat tří další techniky: učení bez učitele, samořízené učení a zpětnovazební učení. Každá z nich má své silné a slabé stránky, které budou dále stručně popsány, a to s cílem vytvoření lepšího přehledu o taxonomii strojového učení.

Učení s učitelem

Učení s učitelem (*Supervised Learning*) – jedná se o dominantní odvětví strojového učení [18], jehož primární úlohou je během procesu trénování naučit budoucí model provádět správné mapování vstupních dat obsahujících individuální příznaky na množinu výstupních cílů (anotací) [17]. V případě aplikování daného způsobu učení je množina tzv. anotací, učícímu se algoritmu dopředu známa. V dnešní době je daná technika velmi populární a to z takového důvodu, že umožňuje natrénovat model s velmi precizními klasifikačními schopnostmi, tj. takovými, při kterých výsledná pravděpodobnost správného odhadu cílové anotace přesahuje 90 %. Do nevýhod daného přístupu trénování je možné zařadit velké nároky na vstupní data. Kromě klasických požadavků spočívající v existenci třech navzájem disjunktivních datových sad pro provedení trénování, validace a testování algoritmu, je zároveň nutné ke každé množině poskytnout množinu namapovaných na ní výstupních cílů (anotací). Je zřejmé, že tuto podmínku není možné splnit u všech druhů problémů [19]. Existují avšak ty, které umožňuje splnit výše zmíněné nároky na vstupní data. Jsou to například binární klasifikace, optické rozpoznání znaků a řeči, klasifikace do několika tříd, predikce syntaktického stromu, detekce objektů, detailněji viz [18, 28].

Učení bez učitele

Učení bez učitele (*Unsupervised Learning*) – opačný pohled na zapojení metod strojového učení k vyřešení různorodých problémů. Jedná se o metodu, v rámci níž neexistuje žádný řídicí prvek, který by prováděl řízení nad procesem mapování vstupních dat na výstupní anotace [19]. Výhodou daného přístupu jsou menší nároky na vstupní datovou sadu. Zmíněnou techniku je možné aplikovat na vyřešení problémů, jejichž složitost a původ neumožňuje jednoznačně vytvořit cílovou množinu anotací. Tento způsob učení je vhodný v případě, kdy vzniká potřeba vytvoření lepšího přehledu o zkoumané datové sadě. V případě zapojení techniky učení bez učitele ještě

před samotným vyřešením definovaného problému (například klasifikační úlohy) lze budoucí model lépe připravit a následně v kombinací s dalšími druhy učení dosáhnout výrazně lepších výsledků. Jako příklad problému, který je možné pomocí dané kombinace technik vyřešit, je shlukování a redukce dimenzionality, viz [18, 28].

Samořízené učení

Samořízené učení (*Self-supervised Learning*) – jedná se o princip učení, které vzniklo na podobných myšlenkách, jak zmíněné výše učení s učitelem (*supervised learning*) [19]. Specifickou vlastností daného přístupu je to, že řídicí prvek provádí dohled nad procesem učení automaticky, tj. bez přímého zásahu programátora. Jelikož se jedná o specifický případ učení s učitelem, zde také existuje označení výstupních cílů (anotací), na které budou v budoucnu namapována vstupní data. Generování anotací probíhá bez zásahu člověka na základě vstupů, například pomocí aplikování heuristických algoritmů [18]. Silnou stránkou daných technik je schopnost učit se bez přímého zásahu člověka do procesu učení. Jako příklad vhodného aplikování principů samořízeného učení je možné uvést predikování dalšího snímku videa na základě existující sekvenci snímků v kodéru, predikci následujícího slova za základě předchozího kontextu (konkrétně se jedná o případ *temporarlly supervised learning*), vytvoření autoenkodérů schopných generovat výstupní anotace z nemodifikovaných vstupních dat apod., detailněji viz literaturu [18, 28].

Zpětnovazební učení

Zpětnovazební učení (*Reinforcement Learning*) – jedná se o velmi zajímavé odvětví strojového učení, které je často součástí vědeckých výzkumů věnovaných pokročilým metodám umělé inteligence a zkoumání psychologie chování jedinců. V odborné literatuře je také možné potkat se s pojmenováním „zpětnovazební učení“ [18]. Zpětnovazební učení dosáhlo svoji popularitu zájmena kvůli úspěchu aplikace *Google DeepMind*, která dokázala naučit se hrát počítačové hry (například *Atari* a také *Go*) líp, něž profesionální hráči [19]. V samotném základu dané techniky leží myšlenka, že jsou agenti (učící se prvky) umisťovány do sledovaného prostředí ze kterého nepřetržitě získávají zpětnou vazbu o vyskytujících se v něm procesech. Identifikovat procesy v prostředí jim nejčastěji pomáhá existující v systému lektor (učitel). Následně, na základě přijatých a rozpoznaných informací, agenti s prostředím interagují a za každou provedenou akci získávají určitou výši odměny. Primárním cílem učícího se algoritmu je tedy maximalizovat svoje finální skóre, a proto se agenti snaží přijaté vstupní informace správně interpretovat a následně vykonat takové akce, které jim umožní získat co nejvyšší výši odměny. [19]

2.4 Zpracování textu pomocí umělé inteligence

Rozpoznání textu a přirozeného jazyka pomocí výpočetních systémů není triviální záležitost. Jedná se o problém zpracování přirozeného jazyka NLP, neboli *Natural language processing*. Jelikož jsou elektronické zprávy nejčastěji založeny na uspořádaném textovém obsahu, který je strukturálně rozdělen do vět a odstavců (sekvence znaků anebo sekvence slov), je možné provádět analýzu přijímaného odkazu a následně provádět jeho klasifikaci, podrobněji viz podkapitola číslo 1.3. Daná technika podle provedených vědeckých výzkumů, viz odborná literatura [41], může být úspěšně aplikována pro rozpoznání a klasifikaci nedůvěryhodných zpráv.

Pokud se detailně zaměříme na proces NLP, dojdeme k závěru, že se jedná o odvětví ve kterém dochází ke sloučení jazykové vědy (lingvistiky) a umělé inteligence. Přičemž, v případě provedení lingvistické analýzy textu není dostačující rozumět významu jednotlivých slov, protože význam samotného slova se silně závisí na tom v jakém kontextu slovo vyskytuje. Samotné odvětví počítačového porozumění přirozenému jazyku vzniklo docela dávno, a proto algoritmů a metod umožňujících provádět porozumění textovému obsahu strojem existuje velké množství. Jako příklad je možné uvést několik problémů, jejichž vyřešení bezprostředně souvisí s NLP. Jsou to například generování textu na základě předchozích slov, odpovídání na otázky související s textem (*question answering*), generování automatických překladů, dolování znalosti z textů, rozpoznání emocí v textu, klasifikace textového obsahu apod.

Klíčová změna ve světě problémů NLP nastala se vznikem hlubokých rekurentních neuronových sítí RRN (*Recurrent neural network*), v jejichž základu ležely buňky umožňující analyzovat textové sekvence vyskytující v minulosti, detailněji viz podkapitola číslo 3.3. Daný přístup umožnil naučit modely provádět analýzu delších textových sekvencí (věty, odstavce). V dnešní době existuje velké množství různých umělých architektur, které vytváří nadstavbu nad klasickými rekurentními sítěmi a dosahují pozoruhodných výsledků [64]. Do takových architektur je možné zařadit `LSTM` (*Long-Short Term Memory*) sítě, `GRU` (*Gated Recurrent Unit*) a další. Každopádně je důležité si pamatovat, že žádný z existujících na moment napsání této diplomové práce algoritmus neumožňuje dosáhnout absolutního chápání jazyka, tzn. porozumění jej na úrovni lidského vnímání [19]. Jedná se o techniky, které zjednodušeně řečeno provádí mapování statistické struktury jazyka a umožňují provést rozpoznání vzorů aplikovaným na textové sekvence [18]. Po vytvoření výše zmíněného mapování jsou umělé sítě schopny řešit lingvistické úkoly.

Vstupní vrstvy neuronových sítí neumí pracovat se slovy v jejich surové podobě, a proto před samotným napojením textové sekvence na neuronovou síť probíhá proces předzpracování dat do číselné sekvence, tzv. numerických tenzorů [18]. Danému procesu se říká *vektorizace* textu a je ji možné docílit třemi následujícími způsoby:

1. Rozdělení vstupní textovou sekvenci na slova a převést je na vektory;
2. Na základě významu vstupní sekvence provést její rozdělení na N-gramy, které budou v dalších krocích převedeny na vektory (*proces vektorizace N-gramů*).
3. Rozdělení vstupné sekvence na slova anebo znaky, které budou vektorizovány v dalších krocích algoritmu;

Jednotlivým častém, které jsou v prvních krocích procesu vektorizace vytvořeny, se říká tokeny a procesu rozdělení textové sekvence na dílčí části tokenizace (*tokenization*). Jednotlivé tokeny jsou namapovány na numerické vektory a napojeny na vstup neuronové sítě v podobě sekvenci tenzorů [19]. Samotný způsob vyřešení problému namapování tokenů na vektory je možné realizovat různými způsoby, v praxi se nejčastěji používají následující dva:

- **Kódování 1-z-n** (*One-hot encoding*) – nejzákladnější a nejjednodušší způsob namapování tokenů na vektory, který spočívá v přirážení tokenům jejich číselných hodnot na základě jejich pozice (indexu) v používaném slovníku [18]. Může být realizována jednoduchým hašovacím (*hash*) algoritmem. Slovník je nejčastěji sestaven ze seznamu nejpoužívanějších slov v datové sadě a setříděn sestupně na základě slovní frekvence.
- **Vnoření slov** (*Word Embeddings*) - velmi efektivní způsob provedení namapování, oproti předchozímu způsobu jsou výsledné numerické vektory mnohodimenzionální (neobsahují prázdné binární hodnoty), tzv. hustá reprezentace tokenů [19]. Docílit vytvoření těchto vazeb je možné buď pomocí speciální vrstvy, která se postupně učí spolu s hlavním modelem a geometricky odráží sémantickou propojenost blízkých slov, anebo pomocí použití univerzálního a vytvořeného předem prostoru vnoření, například pomocí použití knihoven `Word2Vec` anebo `GloVe`.

V současné době došlo k vytvoření vyspělejších způsobů řešení úkolu ze skupiny NLP. Tyto metody hlubokého učení umožňují dosahovat ještě vyšších přesností a provádět dokonalejší mapování tokenů na vektory se zachováním sémantického významu. Navíc se jedná o multifunkcionální modely, tzn. je možné po provedení jejich dotrénování nasadit na vyřešení různých NLP problémů. Do takových technik patří: ULMFiT, BERT, ERNIE, XLNet, RoBERTa, GPT, GPT-2 a další.

ULMFiT

Technika ULMFiT byla vytvořena zakladatelem knihovny `fast.ai` Jeremy Horwardem a vědeckým pracovníkem společnosti DeepMind – Sebastian Ruderem. Jedná se o multifunkcionální generalizovaný model předtrénované neuronové sítě. Předtrénování bylo provedeno na rozsáhlé datové sadě `Wikitext-103`, kterou je možné přizpůsobit na vyřešení skoro jakékoliv úlohy ze skupiny problémů porozumění

přirozenému jazyku, neboli NLP. Proces přeneseného učení spočívá v převodu univerzálního jazykového modelu ULMFiT na doménu konkrétního NLP problému (například problému klasifikace nevyžádaných zpráv). Daný krok probíhá pomocí přeneseného učení poskytovaného modelu na základě textové datové sady, která konkrétní NLP problém dostatečně popisuje. Po provedení přizpůsobení univerzálního jazykového modelu je pomocí metody ULMFiT možné dosáhnout vytvoření textového klasifikátoru, detailněji viz [41].

Google BERT

Jedná se o otevřený framework BERT (*Bidirectional Encoder Representations from Transformers*), který byl vytvořen na základě zveřejněných výsledků řešení různých NLP problémů společností Google. Stejně jako v případě výše zmíněné metody ULMFiT se jedná o generalizovaný jazykový model, který je možné přizpůsobit na velké spektrum NLP problémů. Základní předtrénování bylo provedeno na výpočetních zařízeních společnosti Google. V základním principu BERT leží provedení analýzy vstupního textu z obou stran (zleva a zprava), což ve výsledku umožňuje vytvořit přesnější sémantickou mapu vstupů. Všechny dřívější prezentované *state-of-the-art* modely prováděly analýzu slovních sekvencí pouze z jedné strany. Další zajímavou vlastností je současná multifunkcionalita daného jazykového modelu. V praxi to znamená, že natrénovaný model BERT je možné používat pro současně řešení hned několika NLP problémů. V době svého zveřejnění tento model demonstroval nejvyšší finální výsledky v jedenáctí nejpopulárnějších NLP problémech. Na konci roku 2019 byl BERT připraven na aplikování v 70 různých jazycích. K dalším výhodám daného modelu je možné zařadit jeho rychlé učení oproti klasickým přístupům.

Open AI GPT-2

Velmi známý hráč na poli předtrénovaných modelů schopných řešit úlohy na rozpoznání přirozeného jazyka NLP. Svou popularitu daný model získal tím, že při svém vzniku jeho zdrojový kód byl utajen, protože podle sdílení vývojáře daného modelu se jednalo o natolik inteligentní řešení, které by mohlo ve špatných rukou poškodit nepřipravenou internetovou společnost na tak pokročilé techniky útoků. Samotný model sítě byl vyvinut společností Open AI na obrovské datové sadě obsahující 40 GB textových dat získaných z více něž 8 milionů webových stránek. Nakonec po tak odvážném vyjádření ze strany společnosti Open AI byla zveřejněna zjednodušená verze jazykového modelu obsahující pouze 117 milionů parametrů, oproti 1.5 miliardám v modelu utajeném, detailněji viz [46].

Google Brain XLNet

Jedná se o model vyvinutý týmem společnosti Google Brain zabývající se výzkumy v odvětví strojového učení a umělé inteligence. Neuronová síť `XLNet` (*Generalized autoregressive pretraining method*) stejně jako `BERT` implementuje architekturu transformerů, konkrétně její kodér část. Při svém vzniku síť `XLNet` porazila `BERT` hned v 20 NLP úlohách a dosáhla *state-of-the-art* výsledků v 18 z nich. Oproti `BERT` nezkoumá vstupní sekvenci z obou stran, ale generuje různé permutace vstupních tokenů. Primárně model vychází ze sítě `Transformer-XL` a přebírá z něj dobrou vlastnost ve zpracování delších vstupů. Do klíčových mechanismů `XLNet` umožňujícími `XLNet` realizovat *permutation language modeling* lze zařadit mechanismus pozornosti pro sekvenci obsahů (*content stream attention*) a mechanismus pozornosti pro sekvenci dotazů (*query stream attention*). `XLNet` byl předtrénován na velké datové sadě zahrnující `BooksCorpus`, `EnglishWikipedia`, `Giga5` a `ClueWeb 2012-B`. Navíc fáze předtrénování daného modelu trvala více, než 2000 GPU dnů, viz dále.

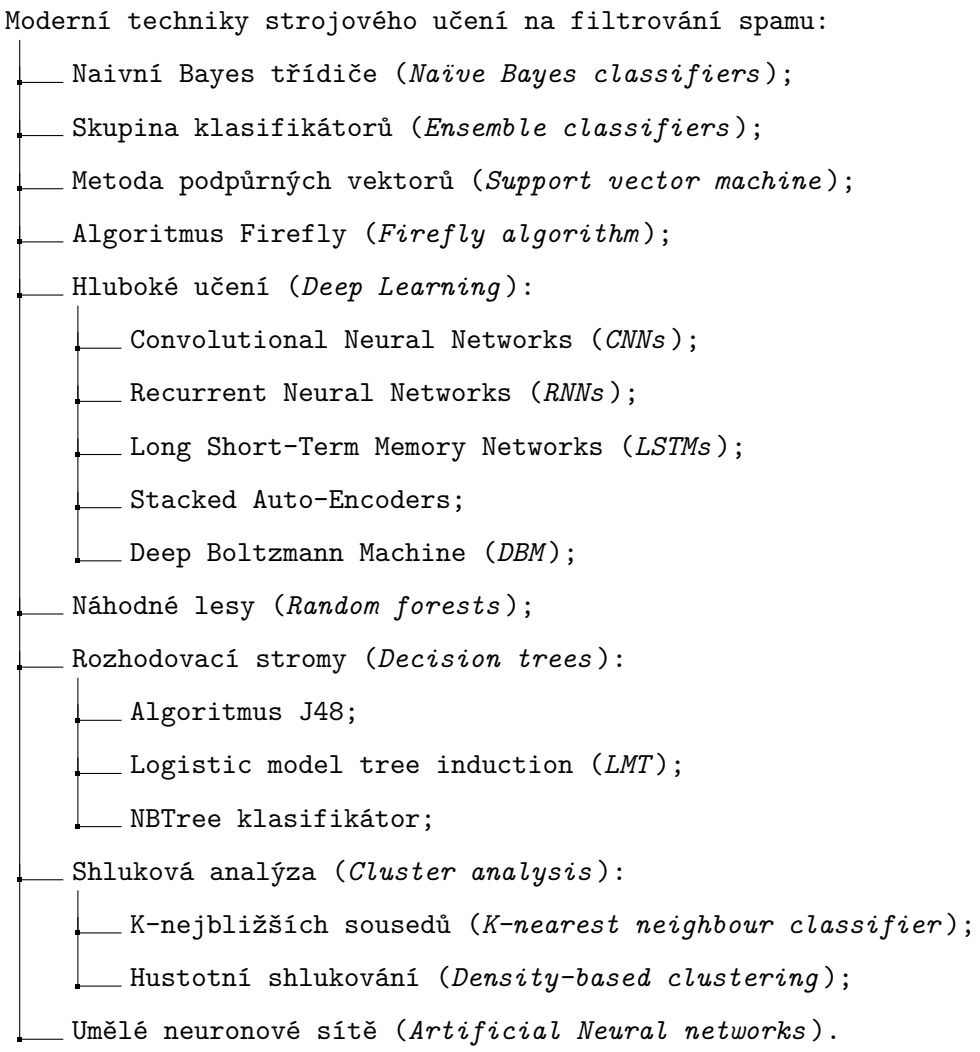
2.5 Současný stav vědy a techniky ve filtrování spamu

Po seznámení s problematikou zpracování textového obsahu pomocí metod umělé inteligence a technik hlubokého učení je možné rozšířit kapitolu číslo 1.5 a detailněji se zaměřit na v dnešní době existující modely strojového učení, které je možné úspěšně aplikovat na provedení analýzy textového obsahu s účelem odhalení spamu.

Během analýzy současného stavu vědy a techniky v oboru informační bezpečnosti, konkrétně v oblasti detekování a filtrace spamového obsahu, bylo možné vydimenzovat nejpoužívanější z existujících přístupů a následně je sloučit do schématu uvedeném na obrázku číslo 2.4 [11]. Je možné si všimnout, že některé z těchto metod se nejčastěji aplikují v rámci textové analýzy (viz podkapitola číslo 1.5), přičemž vybrané z nich mají velký potenciál jak při analýze multimediálních dat (*multimedia content-based*), tak i při provedení *reputation-based* analýzy.

Není možné jednoznačně určit vítěze ve skupině výše uvedených technik strojového učení, protože vhodnost aplikování konkrétní metody se primárně odvíjí od požadavků kladených na filtrační systém, sféry použití a také od specifik konkrétní aplikace. Pokud se přemístíme na doménu filtrování spamu, tak tendence posledních několika let[1] ukazuje, že ve vědeckém světě dochází k přesunu pozornosti z v praxi ověřených technik *Naïve Bayes*, *SVM* a *Decision trees* na metody neuronových sítí.

[1] Vývoj dané tendence je detailně znázorněn a okomentován v článku [11]. V této práci je na stránce číslo 18 (při provedení přímého srovnání existujících metod spamové klasifikace) názorně vidět přenos pozornosti od roku 2016 na odvětví neuronových sítí a hlubokého učení, viz [11].



Obr. 2.4: Moderní techniky strojového učení používané na filtrování spamu

Oproti těmto metodám neuronové sítě ukazují lepší klasifikační schopnosti a jednodušší udržovatelnost z dlouhodobého hlediska [10]. Níže budou v rámci této kapitoly stručně popsány základní principy všech moderních technik strojového učení na filtrování nevyžádaného obsahu. Detailnější přehled je možné dočíst ve zdrojích [10, 11] anebo v jiné odborné literatuře. Podrobnější princip fungování metod umělých neuronových sítí (*artificial neural networks*) a hlubokého učení (*deep learning*) je popsán v rámci kapitoly číslo 2.2 a 3.3.

Naivní Bayes třídiče (*Naïve Bayes classifier*)

Technika Naïve Bayes klasifikátorů je jedna z nejpopulárnějších technik používaných v moderních systémech pro filtrování různorodého textového spamu [11]. Jedná se o metodu strojového učení s učitelem, která vychází z Bayesovy věty a zároveň spadá

do skupiny statistických technik klasifikace [20]. Může být úspěšně aplikována k řešení prediktivních a analytických problémů. Název „Naïve“ u Bayes klasifikátorů pochází z vlastnosti nebrat v potaz šum na vstupu do prediktivní funkce, což ve finále umožní převádět vícedimenzionální problém na skupinu jednodimenzionálních problémů [10]. Pro vytvoření takového klasifikátoru neexistuje jen jeden algoritmus. Pro všechny však platí, že každá vstupní hodnota je zkoumána separátně. V základu Naïve Bayes leží myšlenka detekce populárních spamových slov vyskytujících se v analyzovaném textu, detailněji viz 1.4. Za silnou stránku dané techniky je možné považovat vysokou přesnost[2] a snadnou implementovatelnost. Jelikož se jedná o statistickou metodu, je každé slovo z trénovací datové sady v naučeném modelu vyjádřeno určitou pravděpodobností spadání ke konkrétní klasifikační skupině, tj. *spam* nebo *ham* [11]. Hlavní nevýhoda dané metody vyplývá z dobré odolnosti ke vstupnímu šumu, která na druhou stranu, neumožňuje zachovávat lexikální vztahy mezi slovy v analyzované zprávě, což ve výsledku může negativně ovlivnit finální přesnost klasifikátoru. Detailněji pro tento způsob klasifikace viz ve zdrojích [10, 11].

Skupina klasifikátorů (*Ensemble classifiers*)

Způsob filtrování *ensemble classifiers* je založen na sloučení několika různých klasifikátorů s cílem zvýšení přesnosti finálního systému zaměřeného na detekování spamu [11]. Technika *ensemble classifiers* spadající do skupiny strojového učení během níž dochází k trénování několika klasifikačních modelů pomocí jednoho algoritmu učení [21]. Jedná se o poměrně nový přístup, který v praxi ukazuje svoji efektivitu a dobré generalizační vlastnosti. S cílem pokrytí všech variací spamu, tj. od klasického k pokročilému, jsou skupiny klasifikátorů nejčastěji sestaveny z kombinací různých filtrů [10]. Každý filtr se zaměřuje na určitou cílovou skupinu nevyžádané pošty, což ve výsledku umožňuje dosáhnout dobré odolnosti vůči různorodosti spamových zpráv. Nejpopulárnějšími technikami v *ensemble classifiers* jsou *bagging* a *boosting* [11]. *Bagging* anebo *bootstrap aggregating* se nejčastěji aplikuje na rozhodovací stromy s cílem zmenšit rozptyl klasifikátoru a vytvořit skupiny modelů s různými vlastnostmi z původní datové sady [10]. *Boosting* je iterační technika, která upravuje váhu zkoumaných vstupů v závislosti na výsledku poslední klasifikace [21]. Použití dané techniky umožňuje dynamicky přizpůsobit váhu vstupu v případě nesprávné klasifikace. *Boosting* dovoluje převést skupinu slabých učících se modelů, tzv. *weak learners*, na jeden robustní klasifikační model, detailněji viz [10, 21].

[2]Například v roce 2003 pomocí techniky Naïve Bayes v kombinaci s metodami *binary polynomial hashing* a *orthogonal sparse bigrams* bylo dosaženo přesnosti 99 % ve filtrování e-mailového spamu. Na druhou stranu na úkor časové a paměťové náročnosti algoritmu, viz [20].

Metoda podpůrných vektorů (*Support vector machine*)

Jedná se o techniku strojového učení s učitelem, jejíž pomocí je možné co nejefektivněji[3] dosáhnout dobrých výsledků v případě klasifikace a provedení regresní analýzy [10]. V jádru této techniky leží lineární algoritmus, který je schopen na základě příznaků vstupních dat realizovat rozdělení do dvou klasifikačních tříd. Přiřazení probíhá pomocí jádrové funkce (*kernel function*) provádějící mapování množiny vektorizovaných vstupních příznaků na lineární anebo nelineární příznakovou nadrovinu. Daná nadrovina je ve SVM vytvořena v rámci etapy trénování na základě *initial corpus*. Do základní charakteristiky nadroviny patří vlastnost optimálního rozdělení dvou klasifikovaných tříd, a to takovým způsobem, aby hodnota minimálních vzdáleností jejich bodů byla co nejvyšší. Klíčovou vlastností jádrové funkce je provedení tzv. jádrové transformace umožňující vytvořit lineárně separovatelnou úlohu z lineárně neseparovatelné a následně aplikovat optimalizační algoritmus pro nalezení parametrů rozdělující nadroviny. V práci [10] uvedené srovnání existujících metod SVM v kombinaci s jinými technikami ochrany (pomocí vytvoření *social network*) ukazuje dosažení přesnosti v 97 %. Na úkor své preciznosti nemusí být metoda podpůrných vektorů (SVM) oproti jiným klasifikačním metodám tak rychlá, viz [11].

Algoritmus Firefly (*Firefly algorithm*)

Algoritmus Firefly je meta-heuristická optimalizační technika v tomto kontextu inspirována procesy v biologickém světě a patřící do skupiny algoritmů strojového učení [22]. Poprvé byla prezentována v roce 2008 vědcem Xin-She Yangem s primární motivací technického napodobení chování biologického rodu světlušek, přesněji řečeno, jejich komunikačních schopností během procesu kopulace a oznámení stavu nebezpečí [22]. Ačkoliv byl tento algoritmus ve vědeckém světě kritizován kvůli velké podobnosti s technikou optimalizace hejnem částic (*particle swarm optimalization*), práce [11] ukazuje možné aplikování daného algoritmu na provedení klasifikace nevyžádané pošty.

Shluková analýza (*Cluster analysis*)

Technika *cluster analysis* se v oboru strojového učení používá již dlouhou dobu [10]. Z důvodu své vysoké přesnosti a robustnosti našla uplatnění v různých oborech informačních technologií [23]. Primárními doménami použití jsou dolování znalostí (*data mining*), statistická analýza dat, analýza obrazů, komprese dat a další. Hlavní myšlenkou shlukové analýzy je třídění vstupních objektů do skupin (shluků), a to

[3]Efektivita v tomto případě znamená schopnost systému k generalizaci, nároky na velikost trénovacích dat a schopnost zpracování vysoce dimenzionálních dat [10].

takovým způsobem, aby objekty patřící do stejného shluku měly více podobných příznaků než objekty ze zbylých skupin [11]. Technicky ve svém základu shluková analýza nejčastěji používá metodu učení bez učitele (*unsupervised learning*), tj. trénování modelu probíhá na neklasifikovaných a neoznačkovaných datech [10]. Daná vlastnost umožňuje použít tuto techniku taktéž pro klasifikaci datových sad e-mailů na shluk spamu a hamu. Ve své práci z roku 2011 [23], vědci John Whissell a Charles Clarke prezentovali metodu shlukové analýzy, která byla schopná demonstrovat vyšší klasifikační přesnost nevyžádané pošty, než *state-of-the-art* metody založené na principech samořízeného učení. Populární přístupy, které se ve shlukové analýze na filtrování spamu používají jsou K-nejbližších sousedů (*K-nearest neighbour classifier*) a Hustotní shlukování (*Density-based clustering*), detailněji viz [10, 11].

Rozhodovací stromy (*Decision trees*)

Známá metoda strojového učení s učitelem, která se používá pro vyřešení úloh klasifikace a regrese [11]. Její název odráží základní princip fungování, tj. stromovou datovou strukturu, v níž každý vrchol zastupuje krok rozhodování směřující k výsledné klasifikaci [26]. Vnitřní uzly[4] stromu tvoří rozhodovací podmínky ovlivňující hodnotu cílové funkce. Koncové uzly[5], neboli listy stromu, reprezentují možné hodnoty cílové funkce a tvoří množinu řešení, kterou může klasifikační algoritmus nabývat. Proces rozhodování probíhá od kořene stromu a v závislosti na průběhu cílové funkce se postupně zanořuje do hloubky dokud nedosáhne koncového uzlu zastupující finální rozhodnutí [26]. Nejpopulárnějšími typy stromů používanými na filtrování spamu jsou: stromy postavené na algoritmu `J48`, stromy typu `LMT` (*Logistic Model Tree*) [25] a `NBTree` stromy [26]. Algoritmus `J48` se používá pro vytvoření stromových struktur pomocí zapojení statistických technik. Fakticky se jedná o volně dostupnou a vylepšenou verzi známého algoritmu `C4.5`, detailněji viz [11, 24]. Rozhodovací stromy typu `LMT` přináší do svých uzlů techniku logistické regrese (*logistic regression*). Vytvoření logistického regresního modelu (*LR model*) probíhá pro každý uzel pomocí `LogitBoost` algoritmu. Dále jsou uzly rozdělené pomocí algoritmu `C4.5` anebo `J48`, detailněji viz [25]. Silná stránka daného přístupu spočívá ve vysoké přesnosti finálního řešení. Nevýhoda je způsobena vysokými výpočetními nároky algoritmu [11]. Model `NBTree` v sobě kombinuje techniku rozhodovacích stromů a Naïve Bayes, a to takovým způsobem, aby byly maximálně využity výhody obou přístupů [11, 26]. Propojení je dosaženo vytvořením hybridního rozhodovacího stromu, jehož listy obsahují Naïve Bayes klasifikátory. Tento přístup navíc umožňuje efektivně provádět n-ární klasifikaci, tzn. třídit spam anebo ham na skupiny, detailněji viz [26].

[4]Vnitřní uzel stromu je takový uzel stromu, který není ukončením stromu, ani jeho kořenem [17].
[5]Koncový uzel je takový uzel stromu, který neobsahuje žádného potomka [17].

Náhodné lesy (*Random forests*)

Technika strojového učení, která se používá k řešení klasifikačních a regresních problémů [11]. Ve svém základu vychází z rozhodovacích stromů (*decision tree*) a spojuje tyto datové struktury do komplexnějších struktur (*ensemble decision trees*) [27]. Množina všech rozhodovacích stromů, které jsou vytvořeny pro potřebu klasifikace tvoří les [27]. Samotné rozhodovací stromy se přidávají do lesa během učení algoritmu. V rámci výstupu je z lesa vrácen statistický modus[6] (*statistic mode*) tříd stromů. Aplikování daného přístupu je možné jak pro filtrování spamu, tak i pro jeho kategorizaci [10]. Filtrování je realizováno takovým způsobem, že je vektor příznaků vstupní zprávy analyzován všemi stromy v lese [27]. Každý z nich hodnotí náhodné příznaky a přiřadí zprávě odpovídající klasifikační skupinu. Finální rozhodnutí je uskutečněno na základě nejčetnější odpovědi. Při přímém srovnání náhodných lesů s rozhodovacími stromy, vyplývá závěr, že náhodné lesy mají nižší klasifikační odchylku [11]. Daná technika zároveň ukazuje srovnatelné a v určitých případech lepší výkonnostní výsledky, než SVM (*Support Vector Machine*), viz [11]. Další silnou stránkou náhodných lesů je dobrá škálovatelnost a paralelizovatelnost, schopnost zpracovávat velký objem různorodých dat obsahující mnoho příznaků. Do nevýhod lze zařadit vysokou paměťovou náročnost nutnou pro uložení náhodného lesa.

Hluboké učení (*Deep Learning*)

Technika hlubokého učení (*Deep Learning*), jak bylo zmíněno na začátku této kapitoly, spadá do skupiny strojového učení a představuje velmi efektivní a spolehlivý způsob klasifikace nevyžádané pošty. V základu hlubokého učení nejčastěji leží vícevrstvé umělé neuronové sítě, jejichž architektura je navržena takovým způsobem, aby byly impulzy směrující dovnitř sítě ze vstupní vrstvy analyzovány vetším množstvím skrytých vrstev. Dané vylepšení, jak bylo zmíněno v podkapitole číslo 2.2, s sebou přináší možnost vytvoření vyšší úrovně abstrakce a odhalení složitějších závislostí mezi vstupními daty. V důsledku čehož je toto odvětví strojového učení považováno za jedno z nejperspektivnějších, a to i včetně aplikace na doménu spamové detekce.

Množinu algoritmů hlubokého učení na filtrování spamu lze rozdělit podle způsobu učení[7]. Do skupiny technik učení s učitelem (*Supervised Learning*) je možné zařadit[8] konvoluční neuronové sítě (*Convolutional Neural Networks*), rekurentní neuronové sítě (*Recurrent Neural Networks*) a sítě typu LSTM (*Long Short-Term Memory Networks*). Do skupiny technik učení bez učitele (*Unsupervised Learning*) lze

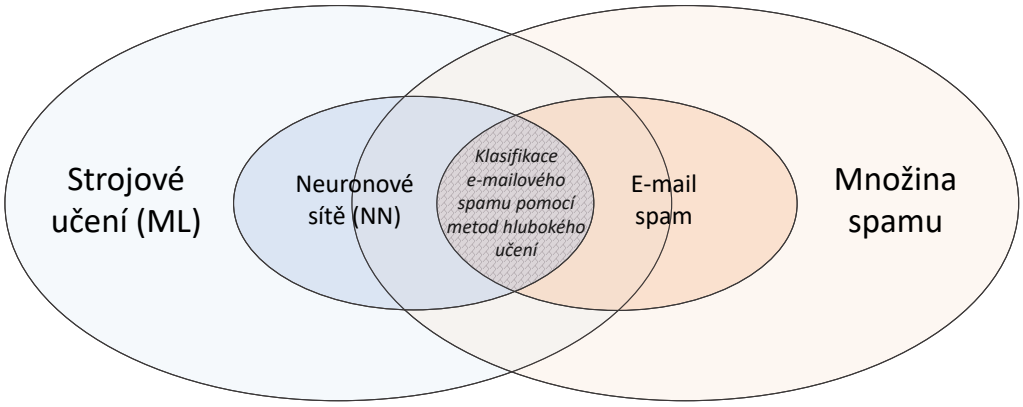
[6]Modus (*Mode*) je ve statistice hodnota, která vyjadřuje prvek s nejvyšší relativní četností ve vybrané množině. Neodborně řečeno je to nejčastěji se vyskytující prvek v množině.

[7]Detailněji je každý způsob učení neuronových sítí popsán v podkapitole číslo 2.3.

[8]Ve vědeckém světě není vyloučená možnost trénování takových sítí i jinými způsoby. Každopádně, při aplikování zmíněného v textu způsobu učení takové sítě demonstrují nejlepší výsledky.

zahrnout auto-enkodéry (*Stacked Autoencoders*), DBM stroj (*Deep Boltzmann Machine*), DBN sítě (*Deep Belief Networks*) a enkodéry typu SSAE (*Stacked Sparse Autoencoder*). Existují také hybridní techniky kombinující dva zmíněné způsoby pro natrénovaní funkčního klasifikátoru. Příkladem může být použití hluboké sítě typu DBN na předtrénování CNN sítě anebo zapojení normalizovaných hlubokých auto-enkodérů do procesu předtrénování a vytvoření přesnějších modelů DNN.

Na obrázku číslo 2.5 je vidět průnik množiny technik strojového učení s množinou zastupující různé druhy nevyžádaného obsahu, neboli spamu. Dané schéma je



Obr. 2.5: Možnosti detekování e-mailového spamu pomocí strojového učení

velmi důležité, a to jak pro pochopení současných tendencí vývoje světa strojového učení, tak i pro lepší orientaci v probírané problematice. Potřeba vytvoření a vysvětlení vzájemných vztahů světů počítačové bezpečnosti a strojového učení vznikla při provedení analýzy současného stavu vědy a techniky ve světě moderních přístupů filtrování spamů založených na hlubokých neuronových sítích. Z obrázku je patrné, že problematika hlubokého učení je podmnožinou umělých neuronových sítí, které jsou ale součástí technik strojového učení. Pokud se zaměříme na doménu spamu, tak zjistíme, že množina nevyžádaného obsahu je složená z různých druhů. Je to například spam na diskusních fórech, spam na internetu, e-mailový spam, multimediální spam atd. Jak bylo zmíněno v kapitole číslo 1, daná diplomová práce se zaměřuje na filtrování e-mailového spamu a proto cílová množina prací vhodných na provedení přímého srovnání s budoucím modelem je tvořena průnikem množiny e-mailového spamu a algoritmů spadajících do množiny neuronových sítí, viz obrázek číslo 2.5.

Primárním účelem provedení analýzy současných řešení bylo vytvoření základové křivky (*base-line*), která by posloužila jako základní měřítko přesnosti a efektivity modelů neuronových sítí natrénovaných v rámci diplomové práce. Jelikož je problematika filtrování spamu poměrně dlouho známá, existuje velké množství vědeckých prací zaměřených jak na vylepšení detekčních schopností současných filtrů, navržení

nových metod s menší mírou *false-positives*, tak i provedení analýzy existujících metod a zjištění *state-of-the-art* výsledků. Specifickou vlastností těchto vědeckých prací zveřejněných v roce 2018–2019 je, že se zaměřují na filtrování e-mailového spamu pomocí metod strojového učení bez cíleného zaměření na techniky hlubokého učení. Navíc, primární důraz je kladen na techniky podpůrných vektorů (*SVM*), náhodných lesů (*RF*) a naivních Bayes třídičů (*Naïve Bayes*), které se při filtrování spamu používají nejčastěji. Zaměření na tyto metody strojového učení přenáší primární soustředění těchto prací na období roku 2004–2016. Ve světě rychlého vývoje technologií neuronových sítí, časová mezera čtyř až pěti let nese nepopiratelný význam a komplikuje přímé porovnání a vytvoření základové křivky (*base-line*). Další komplikace vyplývá z cílů vytyčených v této práci, v nichž bylo rozhodnuto provádět srovnání navrženého modelu s existujícími algoritmy filtrování spamu založenými na technikách hlubokého učení. Z takového důvodu nebylo účelné zahrnovat do srovnání techniky, v jejichž základu neleží hluboké neuronové sítě.

V tabulce číslo 2.1 (viz další stránku) byly ke srovnání zvoleny veřejně dostupné vědecké výsledky za období roku 2016–2020. Všechny práce byly seřazeny na základě data jejich zveřejnění. Důležité je zmínit, že práce byly vybrány takovým způsobem, aby co nejvíce pokrývaly doménu filtrování spamu pomocí analýzy textového obsahu při aplikování technik hlubokého učení. Daný přístup umožní vytvořit co nejkvalitnější základovou křivku (*base-line*) a co nejpřesněji vyhodnotit finální přesnost naučených modelů. Za slabou stránku daného výběru je možné považovat skutečnost, že se ne všechny zvolené práce zaměřují vyloženě na e-mailový spam a ne všechny z nich jsou naučeny na stejných datových sadách[9], což ve výsledku může ovlivnit přesnost hodnocení a působit odchylku. Začlenění těchto technik do výběru bylo argumentováno tím, že se také zaměřovaly na problematiku zpracování přirozeného jazyka (NLP) a binární klasifikaci vstupního obsahu na základě textové analýzy pomocí hlubokého učení.

[9]Je celkem problematické ve vybraných ke srovnání prací odhadnout kvalitu trénovacích a validačních dat, což ve finále může zapříčinit významnou odchylku mezi porovnávanými metodami.

Tab. 2.1: Srovnání moderních metod filtrování spamu pomocí strojového učení

Rok	**Název práce**	**Datová sada**	**Algoritmy**	**Přesnost**
2019	Spam SMS filtering using Recurrent Neural Network and Long Short Term Memory [30]	Spam SMS Collection	RNN a LSTM (Keras a Tensorflow)	98,00 %
2019	Detection of spam review on mobile app stores, evaluation of helpfulness of user reviews and extraction of quality aspects using Machine Learning techniques [31]	Individuálně přizpůsobena App Store datová sada	CNN a CRT, Bi-LSTM spolu s CRF	82,50 %
2019	Opinion Spam Detection in online reviews using Neural Networks [32]	Amazon Mech. Turk, TripAdvisor Review Dataset atd.	CNN v kombinaci s GloVe	88,75 %
2019	Spam detection in social media using Convolutional and Long Short Term Memory Neural Network [33]	SMS spam a Twitter datové sady	LSTM, SLSTM, SSCL, SCNN	99,01 % (SMS) a 95,48 % (Twitter)
2018	Spam filtering using Integrated Distribution-Based Balancing Approach and regularized Deep Neural Networks [34]	Enron e-mail, SpamAssassin, SMS spam	Technika DBB, RDNN-ReL	98,76 % (Enron) a 99,89 % (SA)
2017	Twitter spam detection based on Deep Learning [35]	Datová sada Twitter	Hluboké neuronové sítě (DNN)	94,30 %
2016	E-mail spam classification using hybrid approach of RBF Neural Network and Particle Swarm Optimalization [36]	EmailSpamBase	RBF Neural Networks v kombinaci s PSO algoritmem	93,10 %
2016	Filtering spam using Several Stages Neural Networks [37]	Náhodně zvolená datová sada z SA	Multi-stage Neural Networks	95,82 %
2016	Email spam detection using combination of Particle Swarm Optimization and Artificial Neural Network and Support Vector Machine [38]	Spambase datová sada e-mailů	PSO v kombinaci s ANN a SVM	Hodnota AUC 93,07 %
2016	Content based spam classification - Deep Learning Approach [39]	PU1-PU3, PUA, Enron spam e-mail	Hluboké neuronové sítě	96,21 % (PUA)

V následujícím textu budou vědecké práce uvedené v tabulce číslo 2.1 stručně popsány. Důraz bude učiněn na finální přesnost klasifikátorů, použité techniky hlubokého učení, inovační přínos, silné a slabé stránky modelů atd.

2019 – *Filtrování SMS spamu pomocí RNN a LSTM sítí [30]*:
Autoři článku Arijit Chandra a Sunil Kumar Khatri se zaměřili na filtrování nevyžádaného obsahu šířícího se přes SMS zprávy. Ve své práci popsali existující způsoby detekování a filtrování spamu. Zaměřili se na srovnání technik strojového učení. V praktické části vědecké práce navrhli metodu používající RNN (*Recurrent Neural Network*), konkrétně LSTM (*Long Short Term Memory*) sítě. Pro vytvoření modelů použili volně-dostupnou knihovnu `Keras` a v backendu knihovnu `TensorFlow`. Pro trénování byla použitá veřejná datová sada `SpamSMSColection`, která obsahuje 5574 entit (4827 ham a 747 spam). Předzpracování zahrnovalo odstranění stopwords, tokenizaci a TF-IDF vektorizaci. Výsledná přesnost natrénovaného modelu byla 98 % (13.440s čas kompilace modelu) s možností zlepšení. Navíc byl model porovnán s SVM a Naïve Bayes přístupy, které dosáhli přesnosti 97,81 % a 80,54 %.

2019 – *Detekce spam hodnocení v app store pomocí ML technik [31]*:
Autor Necmiye Genc se ve své práce primárně zaměřuje na posouzení hodnocení uživatelů (*app review*) v obchodu s mobilními aplikacemi pomocí provedení jejich textové analýzy. Prvotním účelem práce bylo vytvoření datové sady hodnocení z cílového obchodu s aplikacemi, a to aplikováním co nejefektivnější techniky. Do dalších cílů patřilo určení v datech parametrů důležitých pro klasifikaci, volba modelů hluboké sítě na vyřešení úlohy, učení a testování modelů, provedení finálního vyhodnocení jejich přesnosti. V rámci dané práce byly na vyřešení postavených úloh (klasifikace, hodnocení užitečnosti) aplikovány tři druhy neuronových sítí: CNN (*Convolutional Neural Network*), Bi-LSTM (*Bi-directional Long-Short Term Memory*) v kombinaci s CRF (*Conditional Random Field*) a DCNN (*Deep Convolutional Neural Networks*) v kombinaci s CRF (*Conditional Random Field*). Finální a nejvyšší přesnost klasifikátoru spamového obsahu byla 82,5 %. Daný model byl naučen na datové sadě obsahující 1800 hodnocení.

2019 – *Detekce spamových hodnocení (online reviews) pomocí NN [32]*:
Autoři Kanagarajah Archchitha a Eugene Yugarajah Andrew Charles se ve svém článku zaměřují na detekci nedůvěryhodných online hodnocení na webových stránkách. Jelikož se jedná o problém zpracování přirozeného jazyka, na jeho vyřešení bylo rozhodnuto použít techniku hlubokého učení, aplikováním CNN (*Convolutional Neural Networks*). Pro dosažení lepších výsledků byla použita univerzální knihovna `GloVe`, která umožnila efektivně provést operaci vnoření slov, tj. *word embedding*. Vytvoření modelu neuronové sítě bylo dosaženo aplikováním knihoven `Keras` a `TensorFlow`. Datové sady pro provedení trénování a testování sítě dohromady obsahovaly 1600 entit. Finální přesnost naučeného modelu byla 88,75 %.

2019 – *Detekce spamu v sociálních sítí pomocí LSTM a CNN sítí [33]*: Práce byla vytvořena autory Gauri Jain, Manisha Sharma a Basant Agarwal. Primárně se zaměřuje na vytvoření modelu hluboké neuronové sítě SSCL (*Sequential Stacked CNN-LSTM*), která bude schopná provádět detekování nevyžádaného textového obsahu na sociálních sítích, např. Facebook, Twitter a také na jiných platformách sloužících pro textovou komunikaci. Architektura modelu byla postavena aplikováním dvou technik, tj. CNN (*Convolutional Neural Networks*) a LSTM (*Long-Short Term Memory*). S cílem dosažení vyšší přesnosti finálního řešení byly použity lexikální databáze sémantických závislosti mezi slovy *WordNet* a *ConceptNet*. Převod textových sekvencí na vektory byl proveden pomocí techniky *word2vec*. Následně byly v rámci práce představeny modely SLSTM (*Semantic Long-Short Term Memory*) a SCNN (*Semantic Convolutional Neural Networks*). Učení modelů bylo provedeno na datové sadě SMS Spam a Twitter Spam. Aktivační funkce byly typu ReLU pro LSTM a Sigmoid pro CNN. Finální přesnost pro filtrování SMS spamu byla následující: SCNN – 98,65 %, SLSTM – 99,01 %, SSCL – 99,01 %. Pro Twitter spam se rovnala: SCNN – 94,40 %, SLSTM – 95,09 %, SSCL – 95,48 %.

2018 – *Filtrování spamu pomocí techniky IDBB a rDNN [34]*: Tento článek byl napsán autory Aliaksandrem Barushkou a Petrem Hajkem. Primární motivací práce bylo vytvoření hluboké neuronové sítě, která by byla schopna překonat hranice přesnosti zadané populárními modely strojového učení, byla odolná vůči přetrénování a schopná efektivně vypracovat vysoce-dimenzionální data. Vyřešení daných problémů bylo možné pomocí jimi navržené techniky N-gram *tf.idf* sloužící k provedení výběru příznaků (*future selection*). Dalším vylepšením bylo použití algoritmu DBB (*Distribution-Based Balancing*) a aktivační funkce typu ReL[10] (*Rectified Linear Unit*). Model umělé neuronové sítě byl postaven na regularizované hluboké síti RDNN (*Regularized Deep Neural Network*). Učení a testování přesnosti modelu probíhalo na čtyřech datových sadách: `Enron`, `SpamAssassin`, `SMS spam` a `SocialNetworking`. Z výše uvedených datových sad je přímo vidět, že byl model testován také na vyhodnocení e-mailového spamu (datová sada `Enron`) a může být přímo porovnán s metodou nabízenou v dané diplomové práci. Nejvyšší dosažená přesnost pro datovou sadu elektronických zpráv `Enron` byla 98,76 % při podílu *features* k N-gram jako 2000/1. Počet entit v datové sadě `Enron`[11] byl 5172 (3672 ham + 1500 spam). Nejvyšší přesnost byla u modelu natrénovaného na datové sadě `SpamAssassin` a rovnala se 99,89 % při podílu *features* k N-gram jako 2000/2. Do nevýhod představených modelů je možné zařadit jejich vysoké výpočetní nároky.

[10] ReL neboli *Rectified Linear Unit* se v odborné literatuře často označuje jako ReLU. Zkrátka ReL byla použitá s cílem, aby co nejvíce reflektovala originální článek [34].

[11] Jedná se o první verzi veřejně dostupné datové sady `Enron`, tj. verzi `Enron 1`. Daná datová sada obsahovala anglické elektronické zprávy vygenerované v organizaci Enron.

2017 – *Detekování textového spamu v Twitter pomocí DNN [35]*:
Autoři článku Tingmin Wu, Shigang Liu, Jun Zhang a Yang Xiang se zaměřili na vytvoření modelu schopného detekovat spam na sociální platformě Twitter. V rámci práce byl zjištěn aktuální stav vědy a techniky a byly porovnány aktuální modely strojového učení. Pokud se zaměříme na prezentované řešení pomocí DNN – provedení mapování syntaxe příspěvků Twitter na vícedimenzionální vektory bylo realizováno pomocí techniky `Word2Vec` a `Doc2Vec`. Pro provedení naučení modelu MLP byla vytvořená rozsáhlá datová sada, obsahující reálné příspěvky z Twitter za deset dnů. Vytvořená datová sada obsahovala 1 376 206 spamových entit a 673 836 důvěryhodných zpráv. Pro provedení testování byla základní datová sada rozdělena do tří menších celků s odlišným poměrem příspěvků spam a ham. Pro všechny datové sady (*Dataset 1 – 3*) byly porovnány přesnosti vybraných modelů strojového učení. Do množiny porovnávaných technik spadly MLP, Naïve Bayes, Random Forest a Decision Tree. Pro datovou sadu s poměrem příspěvku 1:1 (*Dataset 1*) bylo stanoveno, že hluboké neuronové sítě ukazují nejvyšší přesnost, a to ve výši 94,30 %.

2016 – *Klasifikace e-mail spamu pomocí technik RBF NN a PSO [36]*:
Autoři práce Mohammed Awad a Monir Foqaha se zaměřili na filtrování spamu v nejpoužívanějším prostředku komunikace, tj. elektronické poště. V základě prezentovaném v rámci vědecké studie byl použít model neuronové sítě RBFNN (*Radial Basis Function Neural Networks*), který ve svém základu používá aktivační funkci typu RB (*Radial Basis*). Navíc byl pro zvýšení přesnosti sítě použít algoritmus PSO (*Particles Swarm Optimization*), konkrétně varianta HC-RBFPSO. Datová sada e-mailů byla v rámci experimentu stažená z repositáře UCI Machine Learning a obsahovala 4601 e-mailových entit z jedné elektronické schránky, z nichž bylo 2788 zpráv klasifikováno jako spam a 1813 jako ham. Datová sada byla předzpracována a vyčištěna. Maximální přesnost byla dosažená při použití padesáti skrytých neuronů a rovnala se 93,10 %. Do slabých stránek práce je možné zařadit neuvedení hodnot funce *precision* a *recall* při závěrečném testování modelu RBFNN.

2016 – *Filtrování spamu pomocí Several Stages NN [37]*:
Na konci roku 2016 vědci Issa Alkaht a Bassel Khatib zveřejnili článek, v němž byl popsán způsob aplikace sítě typu *Multi-stage Neural Network* na filtraci e-mailového spamu. Zajímavou vlastností daného modelu byla jeho bilingualita, tj. schopnost provádět klasifikaci jako v anglickém, tak i arabském jazyku. Na druhou stranu, byl model neuronové sítě naučen na náhodně vybraných e-mailových datech SpamAssist a jeho velikost pro dva jazyky nebyla dostačující, což ve finále ovlivnilo finální přesnost řešení. Navíc, zdroj [11], kritizuje prezentované řešení za slabý výkon neuronové sítě a nedostatečnou kvalitu dat vybraných na trénování a testování. V nezávislosti na všechny výše uvedené nevýhody, přesnost sítě byla 95,82 % a výkonově byla síť lepší než při aplikování na klasifikaci scén *scenes classification*.

2016 – *Detekce e-mail spamu použitím techniky PSO s ANN a SVM [38]*: Tvůrci článku Mohammad Zavvar, Maysam Rezaei a Shole Garavand se zaměřili na vytvoření hybridního modelu detekce e-mailového spamu, tj. pomocí kombinování různých technik strojového učení. Pro extrakci příznaků z e-mailových zpráv byl použít algoritmus PSO (*Particle Swarm Optimalization*) v kombinaci s ANN sítí. V dalším kroku, pro účely provedení finální klasifikace na dvě podmnožiny, byla využita technika SVM (*Support Vector Machine*). Při testování použita datová sada byla Spambase UCI database. Za slabou stránku dané práce je podle zdroje [11] možné považovat nízký výkon, neuvedení výsledků finální přesnosti modelu a hodnot funkcí *precision* a *recall*. Srovnání bylo provedeno pouze na základě parametru AUC (*Area Under Curve*), jehož hodnota pro demonstrovaný způsob byla 93,07 %.

2016 – *Klasifikace spamu pomocí analýzy kontextu a DNN [39]*: Práce byla vytvořena autorkou Tyagi Akshitou v roce 2016 v rámci projektu Wedge Networks, který zkoumal výkony technik hlubokého učení v odvětví počítačové bezpečnosti. Práce se zaměřovala na filtrování e-mailového spamu pomocí provedení analýzy kontextu aplikováním techniky hlubokého učení SDAE (*Stacked Denoising Autoencoder*), která byla následně porovnána s jinými metodami strojového učení. Primárními modely ke srovnání posloužily hluboké neuronové sítě typu DBN (*Deep Belief Network*), Dense-MLP (*Dense Multi Layer Perceptron*) a nejvýkonnější algoritmy z SVM (*Support Vector Machines*). Vektorizace vstupních slov byla provedená pomocí algoritmu TF-IDF (*Term Frequency - Inverse Document Frequency*). Aktivační funkce pro skryté vrstvy byla *Leaky ReLU* a pro výstupní vrstvy typu *softmax* se ztrátovou funkci typu MCXENT (*Multi-Class Cross Entropy*). Pro natrénování a testování modelu bylo použito pět různých veřejně dostupných datových sad PU1, PU2, PU3, PUA a Enron Spam. Navíc, pro vytvořený model byly spočítány hodnoty *precision* a *recall*. Do nevýhod naučeného modelu je možné zařadit použití zastaralejších datových sad, konkrétně z roku 2000–2006. Nejvyšší přesnost naučeného modelu byla zaznamenána na datové sadě PUA a rovnala se 96,21 %.

3 Metodologie výpočtu a analýza datové sady

Daná kapitola je první, která se věnuje praktické části této diplomové práce. Proces analýzy a přípravy datové sady[1] je nezbytný v případě, jestli chceme natrénovat takovou neuronovou síť, která by dokázala řešit klasifikační úlohu s co nejmenší mírou falešných detekcí. Zároveň se dá říci, že provedení komplexní analýzy a následného předzpracování nakumulovaných dat tvoří tu nejproblematičtější část tvorby funkčního řešení, jelikož vyžaduje dobrý přehled nad zpracovávanými daty a zároveň schopnost odhalit specifické vlastnosti, které nemusí být na první pohled zřejmé.

V moderním světě není problém dostat se k jakýmkoliv datům. Kvůli vzniku moderních komunikačních prostředků a technologii je přirůst dat exponenciální [17]. Daná skutečnost otevírá velmi slibující možnosti pro algoritmy založené na analýze dat a strojovém učení. Problém avšak spočívá v tom, že vznikajícím data nejsou žádným způsobem strukturalizována ani normalizována, a proto největší výzvou pro odborníky v dané sféře je vyfiltrovat pouze taková data, která nesou nejvyšší informační hodnotu a jsou relevantní pro vyřešení konkrétního problému. Zároveň je důležité zmínit, že bez ohledu na to, že neuronové sítě jsou velice perspektivní oblastí strojového učení, stálé nejsou všemocné a vychází pouze z těch dat na kterých byly bezprostředně naučeny. Proto zde stejně jako i v jiných sférách strojového učení bezpodmínečně platí pravidlo, jehož znění je možné zformulovat do následující věty: „Kvalita trénovacích dat na vstupu ovlivní přesnost a spolehlivost systému na výstupu v nezávislosti na tom, jak moderní cílový systém je použit“. Konkrétně to znamená, že v případě, jestli jsou trénovací data nízké kvality, anebo nejsou dostatečně předzpracovány, není účelné očekávat vynikající výsledek, jelikož jsou vstupní data jedním z faktorů finální úspěšnosti [17]. V případě, jestli k procesu předzpracování dat přistoupit zodpovědně, lze dosáhnout mnohem lepších výsledků.

Navíc je nezbytné zmínit, že během provedení jakékoliv analýzy a zpracovávání datových sad není na místě opomínat právní úpravu tohoto odvětví, která klade jistá omezení na práci s daty, detailněji viz podkapitola číslo 3.1.1.

3.1 Charakteristika vstupních datových sad

První podkapitolou a zároveň prvním krokem procesu analýzy datových sad je vytvoření jejich charakteristiky, během níž dojde k vyjmenování všech klíčových specifik dat použitých při učení a validaci vybraných modelů neuronových sítí. Je zřejmé, že pro vytvoření kvalitního řešení s přesností, která by odpovídala moderním metodám filtrování, je nezbytné datum rozumět. Z takového důvodu v podkapitole

[1] Pod pojmem datová sada, anglicky *dataset* anebo *corpus*, je možné představit kolekci entit, které je možné transformovat do strukturované podoby, například tabulky.

číslo 1.2, byla probrána problematika e-mailové komunikace a formátu e-mailových zpráv. Za zmínku také stojí, že natrénováni modelu musí být provedeno na takových datech, která s problematikou filtrování spamu bezprostředně souvisí, tzn. není účelné trénovat síť sloužící na klasifikaci spamových zpráv na datové sadě obsahující výsledky fotbalových zápasů. Taková data z hlediska vyřešení problému klasifikace spamu budou nést nízkou míru informace a lze je přirovnat k náhodným. Z takového důvodu pro natrénování modelu neuronové sítě pro klasifikaci spamu byla zvolena datová sada e-mailů, obsahující jak důvěryhodné, tak i nevyžádané maily. Táto data pochází z různých zdrojů, a proto před provedením všech níže zmíněných kroků byla rozdělena na dvě části. S účelem vytvoření kvalitního modelu neuronové sítě, velikost každého datového rámce je přibližně stejná, tzn. je počet spamových a důvěryhodných e-mailů použitích při učení skoro totožný. Pro lepší porozumění procesu analýzy bude každý z datových rámců zanalyzován a popsán zvlášť, viz další podkapitoly.

3.1.1 Datová sada důvěryhodných e-mailů

V případě vytvoření klasifikátoru spamových mailů je nezbytné kromě spamového datového rámce, poskytnout modelu i množinu důvěryhodných, tzv. čistých e-mailů. Jedná se o elektronické zprávy, které vznikly v rámci reální komunikace mezi lidmi, neobsahující žádné reklamní nabídky, škodlivý zdrojový kód, pokusy o provedení *phishing* útoků atd. Přidání takových zpráv umožní naučit budoucí model tomu, jak vypadají reálné e-maily a reálná komunikace mezi jedinci. Pokud by při pokusu o naučení neuronové sítě nebyl datový rámec čistých e-mailů použit, nejpravděpodobněji by došlo k tomu, že by byl jakýkoliv přijatý e-mail klasifikován jako nevyžádaný, což by ve výsledku zvýšilo míru falešných detekcí, tj. *false-positives* a *false-negatives*.

Největší výzva, kterou bylo nutné při získaní těchto dat vyřešit, spočívala v korektním nakládání s daty vzhledem k právní ochraně e-mailového obsahu. Jelikož každý e-mail je soukromým vyjádřením člověka a zpravidla jej podle něj lze identifikovat, není z pohledu zákona dovoleno jednoduše stáhnout všechny elektronické zprávy uložené na SMTP serveru a provádět nad nimi analýzu anebo výzkum. V případě provedení daného činu dochází k zásahu do soukromí uživatele e-mailové služby, aniž by ten dal k provádění této činnosti souhlas. V takové situaci může dojít k porušení právních předpisů[2]. Použití datového rámce legitimních e-mailů společnosti TrustPort nebylo možné, a to z takového důvodu, že tato množina obsahuje citlivé

[2]Konkrétně se může jednat například o 1) Zákonu č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským; 2) Směrnici Evropského parlamentu a Rady 96/9/ES; 3) Nařízení Evropského parlamentu a Rady (EU) č. 2016/679, plným názvem GDPR (General Data Protection Regulation).

informace, k nímž nebylo možné přistoupit bez souhlasu obou komunikujících stran. Z výše uvedených důvodů byla pro účely této práce použita veřejně dostupná datová sada důvěryhodných e-mailů. Konkrétně se jedná o datovou sadu *Enron Email Dataset*[3], která byla vytvořena v rámci tzv. CALO (*Cognitive Assistant that Learns and Organizes*) projektu[4]. Použitá množina e-mailů obsahuje více něž 500 000 (pět set tisíc) legitimních e-mailů vygenerovaných sto padesáti pracovníky na pozicích *senior management* společnosti Enron. Jedná se o největší existující datovou sadu e-mailů, která je dostupná veřejnosti. Po provedení analýzy veřejné datové sady byly zjištěny informace podstatné pro navazující se zpracování, viz tabulka číslo 3.1.

Tab. 3.1: Popis datové sady důvěryhodných e-mailů

Název datové sady	Enron Email Dataset
Datum zveřejnění	7. května 2015
Autory	Leslie Kaelbling, Malinda Gervasio
Verze	2. verze datové sady
Velikost [MB]	1 330
Formát dat	CSV (*Comma-separated values*)
Komprese	TAR + GZIP
Jazyk	Angličtina
Přílohy	–
Multimediální data	–
Počet souborů [ks]	1
Počet e-mailů [ks]	**517 401**

Na závěr je důležité zmínit, že z hlediska zachování soukromí některých uživatelů, byla datová sada *Enron Email Dataset* v určitém slova smyslu upravena. Konkretně byla provedena řada změn týkajících se osobních údajů v hlavičkách a těle e-mailů. Všechny externí e-mailové adresy v rámci komunikace se zákazníky nepatřící doméně Enron byly odstraněny. Následně došlo k odstranění všech příloh e-mailů (obrázky, binární soubory, textové dokumenty apod). Ve výsledku se jedná o datovou sadu, která obsahuje surovou hlavičku a textovou část (včetně `text/plain` a `text/html`). Dole je možné vidět demonstrativní ukázku staženého CSV souboru.

[3]Datová sada *Enron Email Dataset* je oficiálně dostupna z URL: <`https://www.kaggle.com/wcukierski/enron-email-dataset`>.

[4]Oficiální dokumentace projektu, datové sady a popis poskytované licence je dostupný z URL: <`https://www.cs.cmu.edu/~./enron/`>.

Tab. 3.2: Ukázka datové sady *Enron Email Dataset*

File	Message
allen-p/_sent_mail/1.	Message-ID: <1075855378110.evans@thyme> Date: Mon, 14 May 2001 16:39:00 -0700 (PDT) From: phillip.allen@enron.com To: tim.belden@enron.com Subject: Mime-Version: 1.0 Content-Type: text/plain...
allen-p/_sent_mail/10.	Message-ID: <1075855378456.evans@thyme> Date: Fri, 4 May 2001 13:51:00 -0700 (PDT) From: phillip.allen@enron.com To: john.lavorato@enron.com Subject: Re: Mime-Version: 1.0 Content-Type: text/html...
allen-p/_sent_mail/100.	Message-ID: <1075855687451.evans@thyme> Date: Wed, 18 Oct 2000 03:00:00 -0700 (PDT) From: phillip.allen@enron.com To: leah.arsdall@enron.com Subject: Re: test Mime-Version: 1.0 Content-Type: text/plain...

3.1.2 Datová sada nevyžádaných e-mailů

Jak bylo zmíněno v podkapitole číslo 1.4, podoba spamových e-mailů se neustále vyvíjí, což komplikuje jeho detekci a následující filtrování. Datová sada nevyžádaných e-mailů byla pečlivě vybrána takovým způsobem, aby co nejvíce reprezentovala možné stavy úlohy, tj. podobu různých nedůvěryhodných zpráv během evoluce spamových technik. Daný přístup byl zvolen s cílem vylepšení klasifikačních schopností natrénované sítě. Datová sada spamových e-mailů, která se bude používat během procesu učení, validaci a testování vybraných modelů neuronových sítí, byla ochotně poskytnuta společností TrustPort a obsahuje vzory spamových zpráv za posledních třináct let. Vedení společnosti dalo souhlas na provedení její analýzy a zpracování.

Datová sada byla společností TrustPort nasbíraná za celou dobu fungování společnosti, tj. od roku 2006, až do dneška. Tato datová sada se výrazně liší od datové sady důvěryhodných e-mailů. Hlavní rozdíl spočívá v tom, že *Enron Email Dataset* byla upravena a sjednocená pro co nejjednodušší použití. V případě datové sady společnosti TrustPort se jedná o elektronické zprávy ve své surové, tj. *raw* podobě, které byly zachyceny bezpečnostním síťovým prvkem na perimetru k vnitřní síti společnosti. Konkrétně se jedná o bezpečnostní řešení TrustPort Gateway[5] poskytované

[5]Software TrustPort Gateway je dostupný z URL: <https://www.trustport.com/>.

přímo společností TrustPort. Primárním účelem daného software je provádět analýzu odchozích a příchozích e-mailů. E-maily jsou kontrolovány dvěma bezpečnostními komponentami: Antivirem a Antispamem. Během procesu vyhodnocení analyzovaných e-mailů probíhá přiřazení finálního skóre ověřované zprávě. Čim vyšší je toto skóre, tím pravděpodobnost, že se jedná o spamovou zprávu je vyšší. Po překročení definované hranici je zpráva zablokována a přemístěna do karanténové složky. Důležité je zmínit, že po přesunutí do karantény nebyla spamová zpráva žádným způsobem modifikována, tzn. ani multimediální ani binární přílohy odstraněny nebyly. Bezpečnostní řešení TrustPort Gateway bylo postupně aktualizované, ale místo na ukládání nevyžádaných e-mailů zůstalo stejné, což ve výsledku umožnilo nasbírat velké množství elektronických zpráv za dlouhý časový úsek. Z hlediska metodologie *data mining*[6] se jedná o nesmírně cennou datovou sadu, jejíž pomocí je možné analyzovat tendence vývoje spamových technik, škodlivého kódu a způsobů maskování spamových slov, detailněji viz podkapitola číslo 1.4.

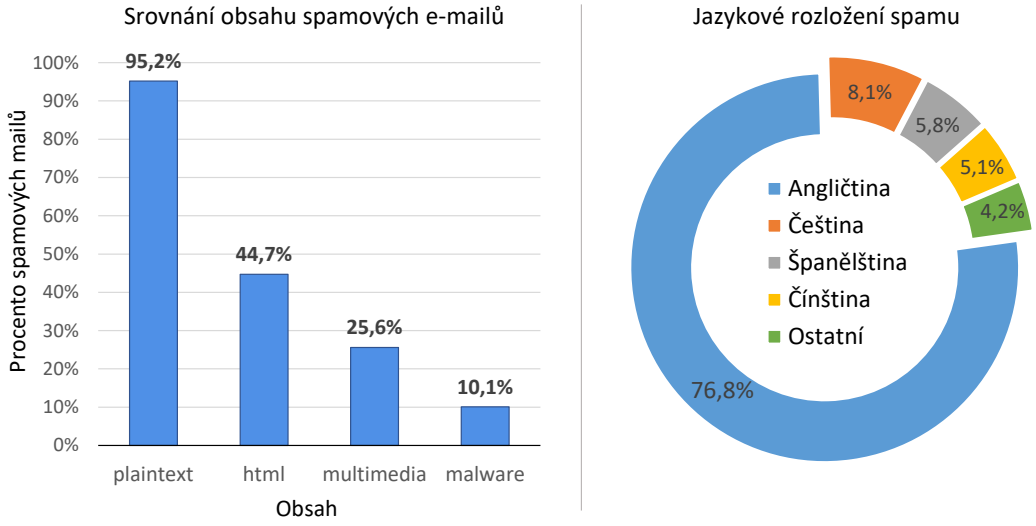
Jak bylo zmíněno výše daná datová sada obsahuje surové e-maily (viz tabulka 3.3), proto při provedení jejich zpracování bude zapotřebí provést složitější přizpůsobení dat, aby byla jejich kvalita na odpovídající úrovni. Jedná se například o odstranění zbytečných položek v hlavičce, klasifikaci jazyků, odstranění příloh, sjednocení a dekódování textových částí (viz podkapitola číslo 3.2).

Tab. 3.3: Popis datové sady spamových e-mailů

Název datové sady	TrustPort Spam Dataset
Datum zveřejnění	interní datová sada
Období	2006–2019 (13 let)
Autory	Roman Vencálek
Verze	1. verze datové sady
Velikost [MB]	11 350
Formát dat	EML, MSG, DAT
Komprese	7z (7-Zip)
Jazyk	Směs různých jazyků
Přílohy	+
Multimediální data	+
Počet souborů [ks]	720 034
Počet e-mailů [ks]	**720 034**

[6] *Data mining*, neboli dolování znalosti, je odvětví, které se zabývá získáváním užitečných a často skrytých informací z velkých objemů dat [17].

Na obrázku číslo 3.1 jsou uvedeny statistiky zohledňující obsahovou část spamových e-mailů. Na grafu vlevo je vidět srovnání obsahu spamových e-mailů. Z kompatibilních důvodů (viz podkapitola 1.2) většina spamových e-mailů (95,2 %) obsahuje část `text/plain`, zbylých 4,8 % jsou zastoupeny takovými nevyžádanými zprávami, které obsahují pouze multimediální část, např. `image/jpg`. Celkové procento multimediálních e-mailů je 25,6 %. Přibližně 44,7 % e-mailů obsahuje část `text/html`, která za sebou může skrývat samotný kód `html` a `css` styly. Následně bylo zjištěno, že 10,1 % mailů obsahovaly škodlivý strojový kód a neprošly na Gateway fází skenování signatur. Na grafu vpravo je vidět jazykové rozložení e-mailů. Největší množství spamových emailů v datové sadě je v angličtině (76,8 %). Danou skutečnost je možné odůvodnit zaměřením společnosti TrustPort na mezinárodní obchod. Zbylou skupinu jazyků zastupuje čeština (8,1 %), španělština (5,8 %), čínština (5,1 %) a ostatní.



Obr. 3.1: Charakteristika datové sady *TrustPort Spam Dataset*

Po provedení detailního srovnání obou datových sad bylo rozhodnuto, že v první řadě budou jako data na učení vybraných neuronových sítí použity všechny spamové emaily v angličtině, které obsahují textovou část. Rozhodnuto bylo pro tento jazyk z takového důvodu, že je zastoupen největším množstvím reálných spamových e-mailů, což v budoucnu umožní vytvořit co nejkvalitnější model. Navíc je celá datová sada důvěryhodných e-mailů také v angličtině viz podkapitola 3.1.1. Proto z důvodu lepšího sjednocení formátu a jazyků obou datových sad budou použity pouze anglické e-maily z datové sady spamových e-mailů viz další kapitola.

3.2 Zpracování dat

Zpracování datových sad, neboli *corpus preprocessing*, je nezbytnou částí procesu vytvoření jakéhokoliv algoritmu založeného na technikách strojového učení. Provedení předzpracování umožní vytvořit takový model sítě, který se bude konstantně přizpůsobovat měnící se podobě spamových technik.

Jak je vidět z popisu datové sady důvěryhodných e-mailů (viz podkapitola číslo 3.1.1) a datové sady nevyžádaných e-mailů (viz podkapitola číslo 3.1.2), jedná se o množinu surových dat, která se liší způsobem uložení, základním předzpracováním a informacemi obsaženými v hlavičce a těle e-mailů. Navíc je důležité vzít v potaz, že v případě čistých e-mailů je k dispozici pouze textová část zpráv a celá tato datová sada je v angličtině. Pokud budeme chtít použit tato data na naučení vybraných neuronových sítí, je nutné provést jejich předpracování a formátové sjednocení s datovou sadou nevyžádaných e-mailů. Proces předzpracování bude vycházet z podoby existujících dat a bude spočívat v sjednocení formátu e-mailů, ve filtraci položek nedůležitých pro rozhodování, v kategorizaci jazyků apod.

V případě, kdy se zaměříme na informace, které se vyskytují v samotných e-mailových zprávách, tak určitě dospějeme k závěru, že některá pole zpráv nenesou pro budoucí model klasifikátoru významnou informační hodnotu. Mohou to například být některá pole hlaviček e-mailů: `Content-type`, `Message-ID`, rodina tzv. `X`-polí apod. Tyto hodnoty se nejčastěji používají pro potřebu SMTP serveru (viz podkapitola číslo 1.2) a slouží pro zajištění správného doručení zpráv. Na základě těchto polí není možné poznat, zda se jedná o vyžádanou zprávu, či nikoliv. Následně, datová sada nedůvěryhodných e-mailů obsahuje multimediální data a binární přílohy se škodlivým kódem. Tento obsah bude nutné při procesu předzpracování vyfiltrovat, a to z důvodu aby byly obě datové sady co nejvíc formátově sjednocené a lišily se pouze textovým obsahem. Celý proces předzpracování dat byl rozdělen do následujících na sebe navazujících kroků:

1. **Základní předzpracování dat** – parsování, dekódováni, oprava chyb;
2. **Čištění dat** – odstranění zbytečných dat a nekvalitních záznamů;
3. **Výběr dat** – zanechávaní relevantních dat;
4. **Integrace dat** – sjednocení formátu datových sad;
5. **Transformace dat** – tagování informací, transformace do vhodné podoby.

Po provedení všech výše zmíněných kroků bude vytvořená datová sada formátově jednotná a připravena pro následující provedení dolování unikátních příznaků, tzn. *feature extraction* pomocí zvolených technik strojového učení.

Je důležité zmínit, že v případě aplikování metod strojového učení, finální příprava vstupujících do sítě dat se bude lišit od zvolené techniky. Z takového důvodu bude finální krok zpracování popsán u každé techniky zvlášť (viz kapitola číslo 3.3).

Pro účely provedení základního zpracování dat byl zvolen vysokoúrovňový skriptovací programovací jazyk `Python`, který je v dnešní době velice rozšířený a dobře zdokumentovaný. Tento programovací jazyk umožňuje efektivně pracovat s velkým objemem komplexních dat a navíc poskytuje externí knihovny umožňující urychlit proces zpracování e-mailů a převedení je do podoby vhodné pro další etapy zpracování. `Python` provádí dynamickou kontrolu datových typu a nabízí objektově orientované a funkcionální programovací paradigma. Navíc celá sada skriptů použitých při zpracování byla nahraná na zálohovací repositář `GitHub`. Jedná se o službu, která poskytuje hosting pro vývoj softwarových aplikací a provedení tzv. *distributed version control*. Výhoda použití této služby spočívá v tom, že umožňuje sledovat historii změn v kódu, vyvíjet projekt paralelně a efektivně provádět vyřešení chyb, tzv. *bugs hunting*. Při napsání skriptů v rámci dané diplomové práce byl důraz kladen na vytvoření co nejrobustnějšího a dobře zdokumentovaného řešení, a to z takového důvodu, aby bylo možné v budoucnu vytvořený kód opětovně používat a provádět efektivní údržbu v nezávislosti na jeho velikosti. Aktuální plně zdokumentovanou a funkční verzi projektu je možné nalézt na oficiálních webových stránkách `GitHub`[7].

3.2.1 Předzpracování dat

Prvním krokem, který byl realizován nad datovou sadou e-mailů, bylo provedení etapy předzpracování dat. Jedná se o nezanedbatelný krok z hlediska zpracování dat, jehož primárním cílem je realizovat přizpůsobení dat do takové podoby, se kterou bude možné v budoucnu efektivně pracovat. Základním úkolem procesu předzpracování dat je vytvoření sady skriptů umožňujících provést parsování e-mailů v jejich surové podobě, dekódování obsahu uložených elektronických zpráv, opravu existujících chyb ve struktuře e-mailové zprávy a načtení e-mailových objektů do operační paměti. Je důležité zmínit, že v případě provedení předzpracování dat pracujeme s každou uloženou e-mailovou entitou zvlášť a postupně, krok po kroku, tak procházíme celou datovou sadu. Jelikož jsou pro vytvoření modelu klasifikátoru použity dvě datové sady odlišných formátů, bude nutné naprogramovat takovou sadu skriptů, která by dokázala tento problém vyřešit a pracovat s daty nezávislé na formátu jejich uložení. Po provedení kroků předzpracování nad jednou e-mailovou entitou úspěšně předzpracovaný e-mail je převeden na objekt a načten do operační paměti parsovacího stroje. E-mailové objekty jsou klasickými instancemi třídy, které mají svůj vnitřní stav a poskytují navenek speciální rozhraní, jehož prostřednictvím je možné přistupovat k interním metodám objektu a měnit jeho vnitřní stav.

[7]Sada skriptů použitých na zpracování datové sady vytvořená v rámci diplomové práce je dostupná na URL: <`https://github.com/ysafonov/SpamDetection`>.

V rámci této etapy došlo k vytvoření univerzálního skriptu, jehož spuštění je možné provést přímo přes příkazovou řádku se specifikací parametrů a vlastnostmi běhu. Ve svém základu skript používá interní knihovnu `Email`[8] s balíky `email.parser` a `email.header`. Tyto balíky umožňuji provádět zakládaní kontrolu a parsování hlavičky e-mailů, dekódovat obsah jednotlivých polí, včetně MIME rozšíření a poskytují funkcionalitu převodu surových bytových objektů na objekty typu `Email`. Druhá knihovna použitá při implementaci této etapy je `Chardet`[9]. Funkcionalita této knihovny byla použita na provedení automatického určení kódování těla zprávy a to v případě, jestli kódování e-mailu není uvedeno v hlavičce e-mailové zprávy, anebo poškozeno. Proces fungování skriptu je možné shrnout do následujících kroků:

1. **Načtení bytového obsahu souboru e-mailu do paměti parseru:**
   V rámci tohoto kroku došlo k přímému načtení e-mailu do paměti parseru. Skript byl vytvořen takovým způsobem, aby byl schopen načítat soubory s rozšířením `DAT`, `EML` anebo přímo z `CSV` formátu. Po úspěšném načtení byl e-mail předán na další zpracování. V případě výskytu chyby se ukazatel parseru posouval na další elektronickou zprávu.
2. **Kontrola e-mailu a převod na `Python` objekt typu `Email`:**
   Po úspěšném načtení byla provedená kontrola polí elektronické zprávy. V případě výskytu závažných syntaktických chyb, byl proveden pokus o opravu anebo parsování pole `Subject` a těla zprávy. Po provedení parsování všech položek prostřednictvím knihovny `Email` byla načtená datová entita převedena na objekt `Python` typu `Email`. Výhoda integrace surového e-mail do objektu typu `Email` spočívá v existenci defaultních metod na ulehčení procesu dekódovaní, přístupu k polím zprávy a iteračnímu procházení její struktury.
3. **Kontrola hlavičky a dekódování pole `Subject`:**
   V tomto kroku byla provedena kontrola hlavičky e-mailu. V případě, kdy pole `Subject` elektronické zprávy bylo zakódováno do znakové sady UTF-8, byla provedena kontrola syntaxe daného pole a jeho dekódování. V případě neexistence předmětu elektronické zprávy byla hodnota pole `Python` objektu nastavena na hodnotu *None*.
4. **Kontrola, určení znakové sady a dekódování těla zprávy:**
   Na základě parametrů zjištěných z hlavičky e-mailu byl proveden pokus o dekódování těla zprávy. Proces ověření použitého kódování je poměrně komplikovaný, a to z takového důvodu, že kromě odlišných znakových sad tělo e-mailu může být kódováno do `base64` formátu. V rámci provedení analýzy robustnosti skriptu bylo provedeno testování na datové sadě spamových e-mailů, které

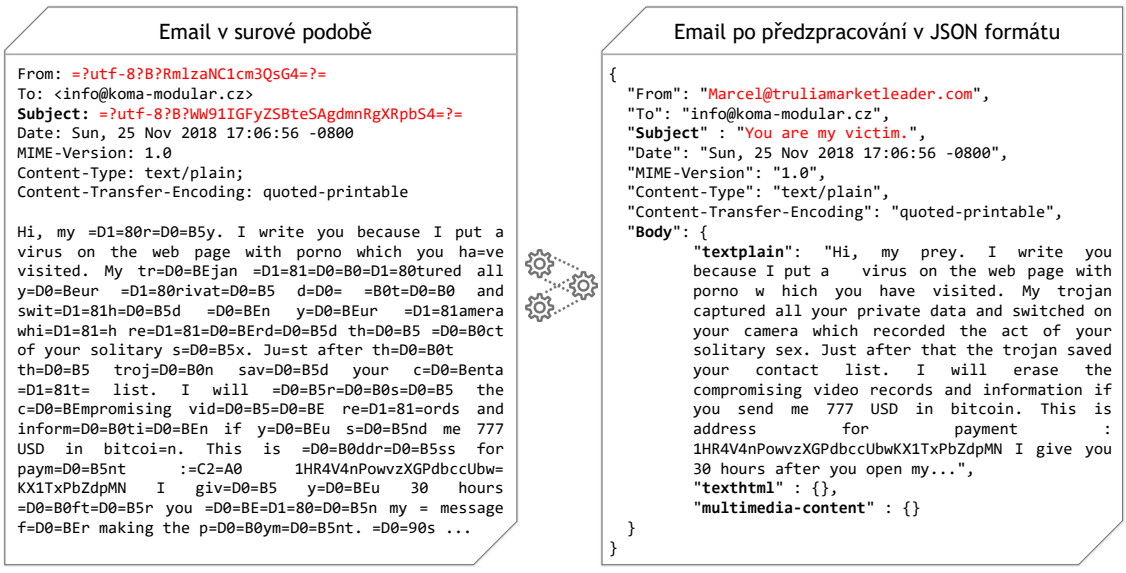
[8]Knihovna `Email` je dostupná hned po instalaci `Python`. Oficiální dokumentaci je možné najít na URL adrese: <`https://docs.python.org/3/library/email.html`>.

[9]Knihovna `Chardet` je dostupna z URL: <`https://pypi.org/project/chardet/`>.

způsobily problémy v bezpečnostním řešení TrustPort Gateway a prošly přes filtr. Po provedení analýzy bylo zjištěno, že velice často útočníci aplikují techniku maskování *obfuscation* znakových sad, anebo používají takové znakové sady, které není možné jednoznačně na straně příjemce dekódovat. V případě výskytu takového maskování může dojít k obejití bezpečnostních mechanismů na straně oběti a doručení spamové zprávy do schránky příjemce. Z takového důvodu při napsání parseru byla použitá externí knihovna `Chardet`, která posloužila na provedení statistického odhadu použité znakové sady s následujícím dekódováním. V případě, kdy nebylo možné dekódovat tělo zprávy ani podle hodnoty pole `Content-type`, ani pomocí knihovny `Chardet`, tak zpráva byla zamítnuta, jelikož byla klasifikována jako nevalidní. Ukazatel parseru se posouval na další e-mail uložený v datové sadě.

5. **Provedení parsování těla elektronické zprávy:**
   V případě, kdy tělo e-mailu bylo dekódované do čitelné podoby, prováděl naprogramovaný skript parsování dekódovaného obsahu e-mailové zprávy. Pokud se položka pole hlavičky `Content-type` rovnala `multipart/alternative`, docházelo k vytvoření objektu `"Body"` a roztřídění do několika kategorií hierarchické struktury e-mailu, např. `textplain`, `texthtml` atd.

Na obrázku číslo 3.2 je možné vidět výsledek předzpracování spamového e-mailu. Vlevo je zobrazena elektronická zpráva v surové podobě, která záměrně neobsahuje specifikaci použité znakové sady. Vpravo je možné vidět výpis předzpracovaného `Python` objektu `Email` ve formátu JSON s hierarchickou strukturou těla zprávy.



Obr. 3.2: Proces předzpracování elektronických zpráv

3.2.2 Čištění dat

V druhé etapě zpracování dat byl důraz kladen na provedení čištění předzpracovaných e-mailových zpráv. Proces čištění spočívá v odstranění nekonzistentních a nekvalitních dat z datové sady. Primární motivací provedení tohoto kroku, je zlepšit kvalitu a relevantnost dat sloužících na naučení modelů neuronových sítí. Zároveň „čistá“ datová sada poskytne lepší základ pro provedení *feature extraction* a tím umožní vytvořit takový model, který bude schopen přizpůsobit se neustálé měnící se podobě spamových zpráv a být efektivní v případě výskytu nového vektoru útoku. Další výhodou provedení daného kroku, spočívá v redukci velikosti výsledné datové sady, co umožní její rychlejší a efektivnější zpracování. V neposlední řadě dojde k odstranění škodlivého šumu tvořeného maskovacími technikami útočníků.

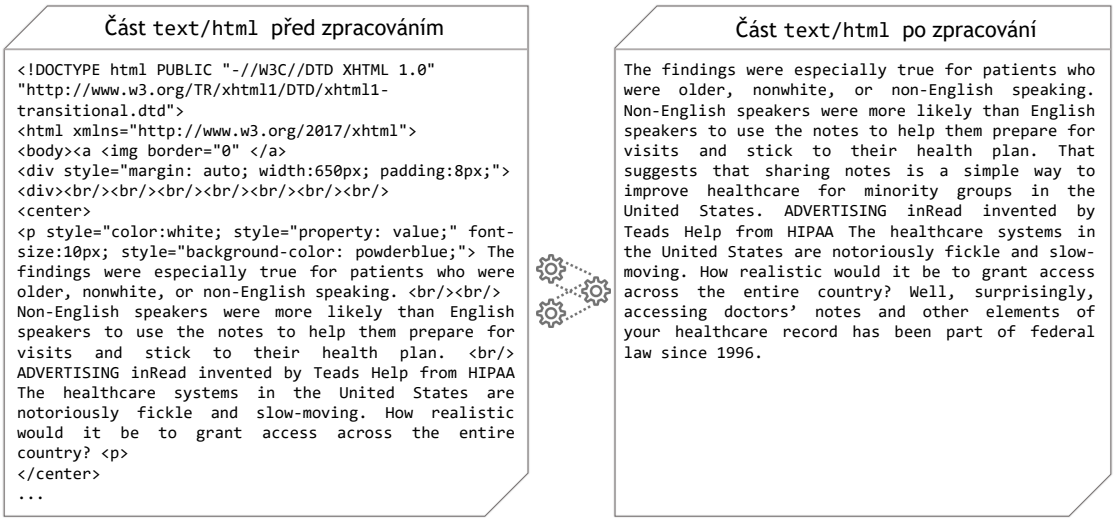
Ačkoliv samotná představa toho, jak má „čistý“ e-mail ve výsledku vypadat je zřejmá, docílení daného stavu je velmi komplikované. A to z důvodu, že není možné předpovědět všechny kombinace spamových e-mailů a také způsoby maskování nevyžádaného obsahu v elektronické zprávě. Po provedení analýzy vybrané datové sady nejsofistikovanějších spamových zpráv z datové sady TrustPort Spam Dataset byla vytvořena sada regulárních *regex* výrazů, které účinně převádí nedůvěryhodný obsah do podoby vhodné pro další zpracování. Bohužel není reálně možné vytvořit univerzální postup a univerzální sadu regulárních výrazů, které by dokázaly poradit s jakýmkoliv textovým obsahem. Za zmínku také stojí skutečnost, že čím vyšší množství regulárních výrazů je na konkrétní obsah aplikováno, tím vyšší bude finální výpočetní složitost algoritmu a tím pomalejší bude zpracování shluku zpráv.

V dané etapě došlo k rozšíření funkcionality skriptu. Do seznamu použitých knihoven byla přidána knihovna `Html2Text`[10] sloužící na odstranění HTML značek (*tags*) z dokumentu a převodu HTML textu na `plaintext`. Aplikování regulárních výrazů posloužilo k odstranění šumu v datech. Jejich účelem bylo hledání symbolů a částí textů, které byly s účelem oklamání spamových filtrů záměrně přidány, viz podkapitola číslo 1.4. Proces čištění dat je možné shrnout do následujících kroků:

1. **Odstranění prázdných e-mailových zpráv:**
   V první řadě došlo k ověření zda e-mailová zpráva neobsahuje prázdné položky, na jejichž základu bude v budoucnu probíhat analýza spamových příznaků. V závislosti na chybějícím obsahu bylo rozhodnuto, zda lze zpracovávanou zprávu doplnit, anebo je lepší provést její odstranění. V případě, jestli zpráva neobsahovala `Subject` bylo rozhodnuto, že táto část bude doplněna začátkem textové částí `plaintext` těla zprávy. Pokud zpráva obsahovala pouze multimediální přílohy anebo binární soubory a nebylo možné doplnit vhodný pro provedení textové analýzy obsah, byla taková zpráva z datové sady odstraněna.

[10] Knihovna `Html2Text` je dostupna z URL: <`https://pypi.org/project/html2text/`>.

2. **Odstranění syntaktických chyb a bezvýznamných symbolů v textu:** Pokud zpráva prošla první etapou kontroly, v dalším kroku bylo provedeno odstranění záměrně udělaných syntaktických chyb buď v samotné struktuře těla e-mailu, anebo v textu. Kontrola byla provedena pomocí použití knihovny `Email`. Odstranění bezvýznamných pro budoucí analýzu symbolu bylo realizováno pomocí regulárních výrazů. Do takových symbolu je možné zařadit netisknutelné znaky, opakující se speciální znaky, zdvojené mezery atd.
3. **Převod stylistického textu na `plaintext`:** V tomto kroku došlo k analýze části `text/html`. Jedná se o část e-mailu, která je posílaná ve formátu HTML. Tento značkovací jazyk byl primárně určen na tvorbu webových stránek s hypertextovými odkazy a navíc umožňuje vložení do své struktury kaskádových CSS stylů. Možnost vložení HTML textu spolu s kaskádovými styly byla přidána e-mailovým zprávám spolu s rozšířením MIME (viz kapitola číslo 1.2). Výhoda daného přístupu spočívá v tom, že je možné do těla zpráv vkládat různorodý HTML obsah a tím zvýšit pohodlí a efektivitu elektronické komunikace. Z hlediska příjemce, se jedná o výhodu, avšak pro účely provedení analýzy textového kontextu, všechny HTML značky tvoří jakýsi šum komplikující provedení vyhodnocení textu a zvyšující redundanci zpráv. S účelem získání co největšího množství informace z existujících e-mailů, bylo provedeno odstranění všech HTML značek. Čištění bylo realizováno pomocí externí knihovny `Html2Text` (viz obrázek číslo 3.3). Odstranění kaskádových stylu bylo provedeno pomocí *regex* výrazů.



Obr. 3.3: Proces čištění textové části e-mailu

3.2.3 Výběr dat

V rámci této etapy došlo k výběru takových dat, která jsou relevantní pro provedení textové analýzy a vyhodnocení elektronických zpráv. Jak bylo uvedeno na začátku kapitoly číslo 3. Hlavním jazykem sady důvěryhodných e-mailů je angličtina a navíc každá e-mailová entita obsahuje pouze textovou část (viz tabulka číslo 3.1). Co se týče datové sady nevyžádaných e-mailů, je tato datová sada tvořena e-maily v různých jazycích. Po provedené analýze bylo zjištěno, že anglické spamové zprávy tvoří 76,8 % ze všech zpráv (viz obrázek číslo 3.1). Navíc, v porovnání s důvěryhodnými e-maily, spamové zprávy obsahují multimediální přílohy a binární soubory.

Z výše uvedených důvodů vznikla potřeba provedení přímé konfrontace datových sad. Po provedení konfrontace obsahu bylo stanoveno, že pro účely textové analýzy budou z hlaviček e-mailů obou datových sad vybrány pouze jejich předměty, tj. pole typu `Subject`. Co se týče samotného těla elektronických zpráv, pro budoucí analýzu budou vybrány části obsahující `text/plain`. Navíc, obsah `text/plain` bude porovnán s částí `text/html`. V případě, jestli se tyto textové části výrazně liší, bude vybraná textová část `text/plain` rozšířena o očištěnou část `text/html`. Celý proces výběru vhodných dat je možné shrnout do následujících kroků:

1. **Odstranění binárních souborů a multimediálních příloh:**
   V rámci prvního kroku došlo k vyjmutí existujících multimediálních příloh a také binárních souborů z těla e-mailu. Realizace daného kroku byla provedena pomocí `Python` knihovny `Email`. Po provedení výběru dat tělo zprávy obsahovalo pouze `text/plain` část a očištěnou `text/html` část.
2. **Analýza a sloučení částí `text/plain` a `text/html`:**
   Po provedení zjednodušení a filtrování těla zprávy, došlo k provedení analýzy a porovnání částí `text/plain` a `text/html`. V případě, že se tyto části obsahově velmi lišily, část `text/plain` byla sloučena s `text/html`. Pokud část `text/html` byla obsahově identická částí `text/plain`, došlo k vynechání oddílu obsahujícího `text/html`.
3. **Extrahování existujících URL odkazů:**
   V rámci zpracování datových sad, bylo provedeno hledání existujících URL odkazů v těle zprávy a jejich duplikování do zvláštní množiny. Tyto odkazy budou použity v budoucnu na provedení pokročilejší analýzy e-mailových zpráv.
4. **Jazyková analýza obsahu e-mailů a provedení filtrování:**
   Během tohoto kroku došlo k výrazné redukci datové sady nedůvěryhodných e-mailů, a to z důvodu provedení jazykové analýzy a odstranění všech jazyků kromě angličtiny. V rámci této etapy byla použitá externí `Python` knihovna `Langdetect`[11]. Výhodou této knihovny je její vysoká přesnost a podpora 55

[11] Knihovna `Langdetect` je dostupna z URL: <`https://pypi.org/project/langdetect/`>.

různých jazyků. Na základě zadaného argumentu knihovna `Langdetect` vrací pravděpodobnostní odhad možného jazyku. Modul `Langdetect` umí provádět výsledný odhad ve dvou režimech: deterministickém a nedeterministickém. Nedeterministický režim znamená, že v případě opakovaného spouštění kódu se stejným vstupním argumentem může dojít ke změně pravděpodobnosti a výstupního jazykového odhadu. Je také důležité zmínit, že pravděpodobnostní odhad přímo závisí na délce ověřovaného textu. Pokud vstupní text obsahuje pouze jedno slovo, tak je pravděpodobnost odhadnutí správného jazyka velmi malá a není možné takovému výsledku jednoznačně důvěřovat. Na obrázku číslo 3.4 je uvedená naprogramována funkce sloužící k provedení jazykové analýzy a detekci požadovaného jazyka. V případě, jestli vstupní text přesahuje minimální počet slov a zároveň byl externí knihovou `Langdetect` rozpoznán za hledaný, funkce vrátí hodnotu `True`. V opačném případě dojde k vrácení `False`. Zároveň je důležité zmínit, že při provedení rozpoznání jazyka zprávy probíhá analýza jak předmětu zprávy, tak i těla. V případě, jestli obě části budou kladně vyhodnoceny, dojde k provedení dalších kroků.

```
1   # Daná statická metoda slouží na vyhodnocení, zda se jedná o vstupní text
2   # hledaného jazyka. Jako vstupní argument metoda přebírá text, zkratku hledaného
3   # jazyka a také pravděpodobnostní hranici, která má být překročena.
4   @staticmethod
5   def is_requiredLanguage(text, language, probability):
6       if text == None or words_count(text) ⩽ Language._minimum_word_count:
7           return False
8       predictedValue = detect(text)[0]
9       if predictedValue.lang == language and predictedValue.prob ⩾ probability:
10          return True
11      return False
```

Obr. 3.4: Výpis statické `Python` metody sloužící pro klasifikaci jazyků

Po provedení všech výše uvedených kroků, byl proces výběru dat u svého logického konce. Ve výsledku obě datové sady obsahovaly pouze anglické e-maily, přičemž každý z nich obsahoval: předmět zprávy, tj. `Subject`, tělo obsahující `plaintext` a také množinu vyskytujících URL odkazů.

3.2.4 Integrace dat

Po provedení zpracování dvou datových sad bylo nutné provést sjednocení jejich formátu a převedení do takové podoby, se kterou bude možné v budoucnu efektivně pracovat a v případě potřeby ji jednoduše modifikovat. Pro tyto účely byl zvolen

formát CSV, neboli *Comma-separated values*. Jedná se o velice flexibilní a jednoduchý formát ukládání dat, který umožňuje převést tabulková data do textové podoby a v případě potřeby je jednoduše načíst znovu. Jednotlivé hodnoty jsou ve formátu CSV uzavřeny do uvozovek a jsou odděleny jedním znakem. Jako oddělovač může posloužit[12] například středník, čárka, tabulátor atd. Oddělení řádků tabulek je realizováno přechodem na nový řádek. V případě, jestli v uloženém textu vyskytují uvozovky, tak z důvodu zachování integrity dat budou výsledné uvozovky zdvojeny. Jako výhodu zvoleného způsobu je možné zmínit minimální redundanci[13] při uložení dat. Stručný popis výsledné struktury CSV dokumentu, který byl vytvořen po provedení zpracování e-mailů, je uveden níže (viz obrázek číslo 3.5).

```
 ID              Subject           Body                          Urls
"13207054898446";"Re: Hi Dear  ... ";"Hi Dear, We produce hot  ...  ";"http://srv ... "
"13207055055763";"Russian girls ... ";"Visit Bride.ru View photos ... ";""
"13207055058899";"Biggest DIRECT ... ";"Discount -40% Customer ... ";"http://cows ... "
"13207055114514";"Salutations ... ";"I was mistreated in the ... ";"http://corners ... "
      ...               ...               ...                           ...
"13207056012414";"Re: Welcome  ... ";"Hi Dear, We produce hot  ...  ";"http://srv ... "
"13207055055432";"Sexy Girls ... ";"Hi, see these amazing photos ... ";"http://se ... "
"13207051211399";"Re: Plastic m ... ";"Discount -80% Go Go ... ";"https://cars/ ... "
"13201211114514";"My dear Roman ... ";"I was waiting for you ... ";"http://horny/ ... "
```

Obr. 3.5: Struktura vygenerovaného CSV dokumentu spamových e-mailů

Z obrázku je patrné, že je CSV dokument zpracovaných elektronických zpráv rozdělen do čtyř sloupců: `ID`, `Subject`, `Body`, `Urls`. Sloupec `ID` obsahuje unikátní identifikátor e-mailové zprávy, který byl přiřazen během zpracování. Sloupce `Subject` a `Body` zastupují hodnoty hlavičky a textové části těla e-mailu. Tyto hodnoty prošly etapami čištění a výběru a obsahují minimální množství šumu. Poslední sloupec `Urls` obsahuje všechny URL odkazy, které byly detekovány během etapy výběru dat, viz podkapitola číslo 3.2.3.

V rámci etapy integrování dvou datových sad došlo k vytvoření obecného přehledu o finálním množství e-mailových zpráv a také o času nutným na provedení všech výše zmíněných operací zpracování surových dat. Výsledné hodnoty je možné vidět v tabulce číslo 3.4. Z tabulky je patrné, že číslo anglických spamových zpráv po filtrování je 553 229. Počet zpráv v datové sadě důvěryhodných e-mailů je 516 957.

[12]Symbol oddělovače použitého v CSV dokumentu výhradně závisí na implementaci algoritmu sloužícího k uložení a načtení CSV souboru.

[13]V porovnání s populárními formáty XML a JSON, obsahují dokumenty CSV výrazně nižší množství režijních symbolů.

Celkové číslo elektronických zpráv vhodných pro aplikování vybraných neuronových sítí je 553 229 + 516 957 = 1 070 186. Zpracování datové sady nedůvěryhodných e-mailů celkem zabralo 18,4 hodin, tzn. průměrná doba nutná na zpracování jedné zprávy byla 0,119 sekund. Z tabulky vychází, že zpracování datové sady důvěryhodných e-mailů zabralo miň času, konkretně 13,6 hodin. Průměrná doba na zpracování jedné zprávy byla také menší a rovnala se 0,096 sekund (viz tabulka číslo 3.4).

Tab. 3.4: Statistika zpracování datových rámců

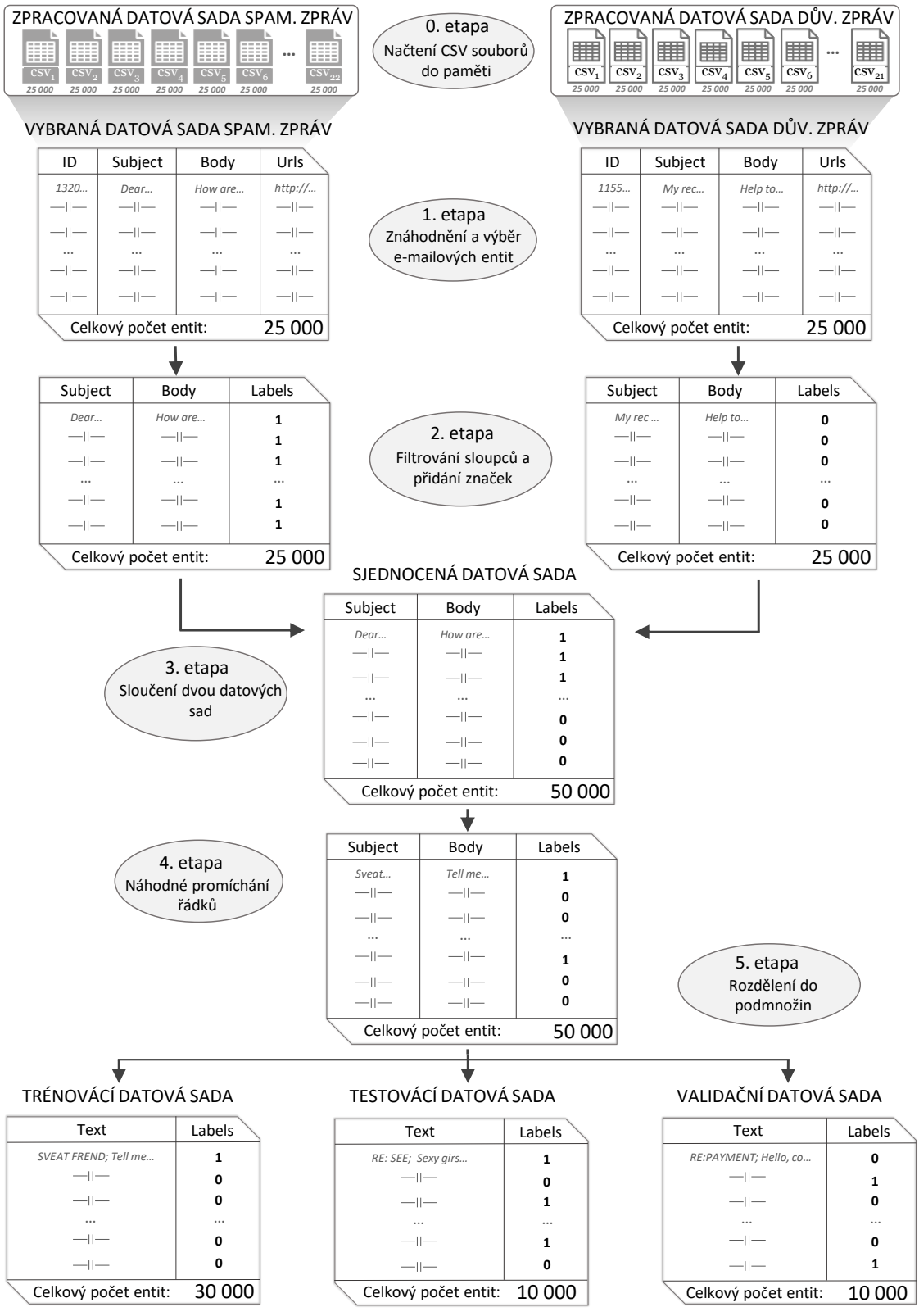
Název datové sady	Průměrný čas zpracování zprávy	Čas zpracování datové sady	Finální počet entit
Enron Email Dataset	0,096 s	13,6 hod	516 957
TrustPort Spam Dataset	0,119 s	18,4 hod	553 229
Součet	-	**32,0 hod**	**1 070 186**

V daném kroku je důležité zmínit o provedeném rozdělení zpracovaných datových sad. Jelikož celkové množství e-mailů převýšilo hranici jeden milion, bylo velmi problematické efektivně pracovat s tak velkým množstvím dat, zejména kvůli rozsáhlosti výsledného CSV souboru[14]. Z takového důvodu bylo rozhodnuto, že při zpracování dvou datových sad výsledný CSV soubor bude rozdělen na menší části obsahující pouze 25 000 entit. Ve výsledku bylo obdrženo 43 CSV dokumentů pokrývající celý rozsah e-mailů vhodných na provedení transformace, viz obrázek číslo 3.6.

3.2.5 Transformace dat

V daném případě byl proces transformace těsně spojen s etapou integrace dat, zejména kvůli tomu, že během této etapy došlo k transformaci dat do takové podoby, aby byla po následujícím finálním přizpůsobení připravena k modelování. Jelikož v případě vytvoření mechanizmu umožnujícímu provádět filtrování spamových zpráv, se jedná o klasifikační problém, který bude probíhat pod dohledem učitele, tj. *supervised learning*, je nutné provést značkování všech e-mailových entit uložených v CSV dokumentech. Přidání značek ke každé elektronické zprávě umožní budoucímu klasifikátoru poznat, o jakou kategorii e-mailů se jedná. V rámci provedení značkování (*tagging*) e-mailových zpráv došlo k přiřazení spamovým e-mailům značky `label=1` a skupině důvěryhodných e-mailů značky `label=0`. Proces transformace obou množin je uveden na obrázku 3.6.

[14]Výsledný surový CSV dokument obsahující datovou sadu čistých e-mailů, tj. *Enron Email Dataset*, bez komprese zabíral na disku více než 2 GB prostoru.



Obr. 3.6: Proces transformace a značkování datových rámců

Proces transformace a značkování datových rámců je možné logicky rozdělit do šesti etap: načtení CSV souborů do paměti (0. etapa), znáhodnění a výběr různorodých e-mailových entit (1. etapa), filtrování duplicitních a prázdných řádků (2. etapa), sloučení vybraných a označkovaných množin e-mailů (3. etapa), náhodné promíchání řádků ve sloučené množině (4. etapa). Po provedení procesu transformace byly úspěšně vygenerovány tři množiny dat, které budou použity během procesu trénování, validace a testování (5. etapa). Zároveň je důležité podrobněji vysvětlit princip výběru e-mailových entit ze zpracovaných datových sad, odůvodnit vytvořená omezení finálního množství vybraných e-mailů po etapě načtení CSV souborů, viz obrázek číslo 3.6. V rámci plnění semestrální práce bylo provedeno zkušební trénování prvního modelu vybrané neuronové sítě. Jeho primárním účelem nebylo dosažení nejvyšších hodnot přesnosti, ale odhalení optimální velikosti vstupní datové sady podle sledování hodnot validační a trénovací ztrátovosti (*loss*). Během daného experimentu bylo zjištěno, že přenesené učení sítě probíhá nejúspěšněji, když vstupní trénovací sada nepřevyšuje 30 000 hodnot. Z takového důvodu bylo rozhodnuto, že pro trénování, validaci a testování modelů budou v rámci diplomové práce použity speciálně vybrané datové rámce. Použití celé datové sady obsahující více než milion entit nebylo nutné, jelikož by při procesu učení sítí docházelo k jejich přetrénování a dosažení nízké přesnosti na testovacích datech. Celkový počet záznamů ve speciálně vybrané datové sadě byl roven 50 000, přičemž polovinu, tj. 25 000 záznamů, tvořily nevyžádané elektronické zprávy. Zbylá polovina obsahovala důvěryhodné e-maily. Princip výběru dat byl realizován takovým způsobem, aby spamová datová sada nasbíraná za třináct let ve společnosti TrustPort byla v redukovaných datových rámcích zastoupena co nejvíce různorodými e-maily, pokrývajícími co nejširší časový úsek. Primárním cílem daného kroku bylo poskytnutí trénovaným modelům co nejširšího pohledu o dynamičnosti vývoje spamových technik. Výběr dat probíhal tak, že po načtení CSV souborů do paměti byl blok spamových e-mailů za roční období náhodně přemíchán a do výstupních datových rámců bylo přeneseno takové množství e-mailů, aby trénovací, validační a testovací množiny splňovaly očekávané velikostní požadavky. Co se týče množiny důvěryhodných e-mailů, princip její výběru byl podobný a zahrnoval náhodné smíchání řádků, odstranění stejných e-mailů apod. Níže budou popsány všechny tři množiny e-mailů, které byly nutné na realizaci fáze učení algoritmů a nezávislého testování přesnosti binární klasifikace.

Trénovací datový rámec

Primárním účelem trénovací datové sady je provést co nejefektivnější nastavení parametrů učícího se algoritmu [17]. Obvykle tato datová sada tvoří největší část původní množiny dat. Po provedené etapě transformace výsledná trénovací datová sada

obsahovala 60 % záznamů z vybrané množiny, což odpovídalo 30 000 záznamům, detailněji viz obrázek číslo 3.7. Pro účely napojení zvolených modelů neuronových sítí bylo zároveň nutné provést sloučení předmětů e-mailových zpráv spolu s jejich textovými částmi. Provedená transformace je uvedena na obrázku číslo 3.6. Je z ní patrné, že `Subject` zprávy byl převeden do velkých písmen (*uppercase*) a navíc je ukončen oddělovačem v podobě středníku, viz obrázky čísla 3.7 a 3.8. Je důležité zmínit, že v případě modelu `BERT` a `XLNet` budou k datům v dalších krocích přidány pomocné značky, a to s cílem rozpoznání struktury vstupních sekvencí sítěmi, viz 3.5 a 3.6. Bude navíc nutné respektovat další požadavky kladené vybranými modely, konkrétně se jedná o schopnost sítě zpracovávat malá a velká písmena, viz dále.

```
================================ TRAIN DATAFRAME ================================
       labels                                               text
0           1  RE: MY BENGALI ; Dropping program and the ...
1           1  FORETOLD MORAL SUPPORT ;  He said the other cl ...
2           1  LABRADORITE FORMALDEHYDE ; after Sunday's expe ...
3           1  IN INDIA, PEOPLE, WOMEN IN PARTICULAR EITHER F ...
4           1  THEMSELVES NEITHER IN. ; Please Help us ! Visi ...
...       ...                                                ...
29995       0  CAISO NOTICE: EXECUTIVE SUMMARY OF STAKEHOLDER ...
29996       0  RE: GIVE UP AGREEMENTS: BANC ONE ; Mary: These ...
29997       1  CONFUSE WARPED ;  41 and expect it to be there ...
29998       0  ALLIANCE NETBACK WORKSHEET ;  Forwarded by Phi ...
29999       0  CLEARSTATION NEWS - JANUARY 2002 ; Happy New Y ...

[30000 rows x 2 columns]
============================== END TRAIN DATAFRAME ==============================
```

Obr. 3.7: Výpis trénovací datové sady

Testovací a validační datové rámce

Následně, po provedení etapy transformace, byly z původní datové sady vygenerovány dva datové rámce stejné velikosti. Oba obsahovaly výběr elektronických zpráv, a to takovým způsobem, aby co nejvíce pokrývaly neredukovanou množinu e-mailů. Každý z datových rámců obsahoval 20 % záznamů ve vztahu k vybrané trénovací množině, což odpovídalo 10 000 záznamům, detailněji viz obrázek číslo 3.8.

Validační datový rámec je bezprostředně zapojen do samotného procesu učení algoritmu a slouží k ověření jeho přesnosti během procesu trénování. Jelikož se částečně používá i pro finální ladění trénovaného systému, není účelné jej aplikovat pro odhalení finální přesnosti učícího se algoritmu. Z takového důvodu je ve světě

strojového učení samozřejmostí vytvářet třetí datový rámec, který je co nejvíce nezávislý na ostatních datech a je zapojen pouze do procesu testování natrénovaného modelu [17]. Na obrázku číslo 3.8 jsou demonstrovány vytvořené testovací a validační datové rámce. Je možné vidět, že struktura vytvořených sad je velmi podobná výše popsáné trénovací množině. Každý datový rámec se skládá ze dvou sloupců. První zastupuje přiražené značky a druhý obsahuje textovou část, která stejně jako u trénovacího rámce používá středník pro odlišení předmětu a těla e-mailové zprávy. Testovací množina e-mailů je vidět v prvním výpisu jako *text dataframe* a validační sada je na obrázku číslo 3.8 označena jako *validation dataframe*.

```
================================ TEST DATAFRAME ================================
      labels                                                 text
0          0  RE: FLAG FOOTBALL ; Saturday is better for me  ...
1          1  BUFFOON TURTLE ;  In this article, we walk thr ...
2          0  DARREL JACKSON ; I want to place a 1 bid for d ...
3          0  RE: WEFA'S OUTLOOK FOR NATURAL GAS ; Thanks Fr ...
4          0  RE: CASH MKTS ; Freak show. started at -7. wen ...
...      ...                                                 ...
9995       0  ASSET MARKETING ; We are pleased to announce t ...
9996       0  RE: GAME TONIGHT ; be there on time beaaaaaccc ...
9997       0  IN LONDON OFFICE - JANUARY 15-19 ; I will be w ...
9998       1  MARKET HOT PERORT. ; Gains symbol report http: ...
9999       0  HEY, CAN GREG FIT ME IN FOR ABOUT 30 MIN ; Hey ...

[10000 rows x 2 columns]
============================== END TEST DATAFRAME ==============================
```

```
============================== VALIDATION DATAFRAME ==============================
      labels                                                 text
0          0  FW: ;  Original Message From: "Matt Cuocci" ma ...
1          1  SPADES ;  Moeller: Actually, you have the AGX3 ...
2          0  LIVELINK ID SETUP CONFIRMATION NOTIFICATION ;  ...
3          1  RE: IGNORE SYRINX ; Of five funerals began Fri ...
4          0  RE: ; Sounds as if you went to a better show t ...
...      ...                                                 ...
9995       1  IMPORTANT COMMUNIQUE. ; Immense gains in Germa ...
9996       1  IN FACT, SOMETIMES, THAT'S A GOOD THING! ;  Th ...
9997       1  TRACTION YIPPEE ;  WIND ADVISORY IN EFFECT FRO ...
9998       1  HIGH TIDE SHARPENER ;  Freeware: iConcertCalTh ...
9999       0  RE: DRAFT ; alright, no more memos about bw3 s ...

[10000 rows x 2 columns]
============================ END VALIDATION DATAFRAME ============================
```

Obr. 3.8: Výpis testovacího (*test*) a validačního (*validation*) datového rámce

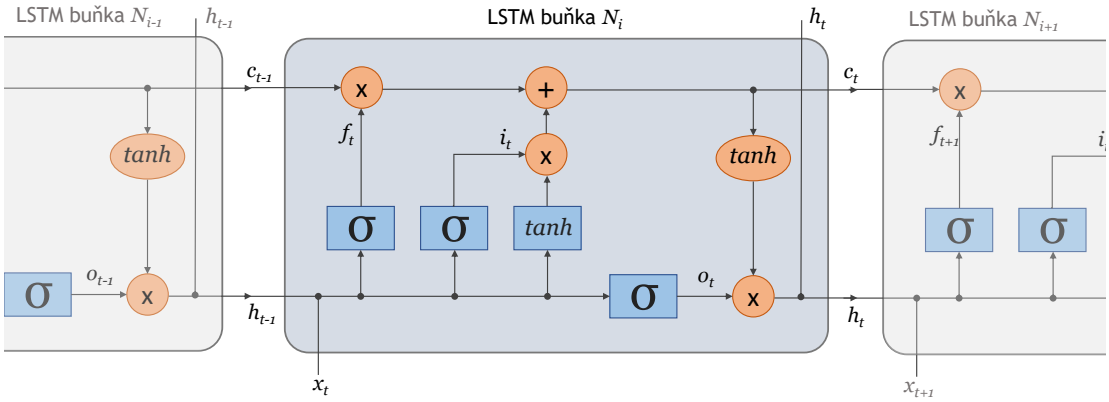
3.3 Volba modelů neuronových sítí a jejich specifika

3.3.1 Model ULMFiT

Charakteristika modelu

První technika, která byla v rámci diplomové práce na problematiku filtrování spamových zpráv pomocí umělé inteligence byla ULMFiT (*Universal Language Model Fine-Tuning*) [41]. Jedná se o *state-of-the-art* metodu, která se aplikuje na vyřešení problému porozumění přirozenému jazyku, neboli NLP (*Natural Language Processing*). Jelikož problematika textové klasifikace spadá pod skupinu NLP problémů, zvolenou metodu bylo možné úspěšně aplikovat i klasifikaci spamových zpráv [41, 64, 62]. V základu techniky ULMFiT leží myšlenka klasických LSTM (*Long Short-Term Memory*) vrstev, které hodí na zpracování textových dat, jelikož umožňuji „kódovat“ do struktury vnitřních vrstev neuronové sítě ne pouze vyznám jednotlivých slov jdoucích za sebou, ale mnohem komplexnějších sekvence, jako jsou například věty a odstavce. V samotném základu LSTM sítí leží myšlenka rekurentních neuronových sítí RNN (*Recurrent neural network*), které pomocí specifického propojení skrytých neuronů umožňují zpracovávat a hledat závislosti mezi daty, které prošly vrstvou v minulosti [18]. Existuje avšak problém, který klasická implementace rekurentních neuronových sítí přímo řešit neumí. Principiálně se daný problém projevuje během trénování rekurentních sítí a spočívá v neschopnosti sítě provádět analýzu dlouhodobých časových závislostí mezi vstupními daty [18]. Danému efektu se říká problém mizení gradientu (*vanishing gradient problem*), který pokouší vyřešit komplexnější LSTM struktury. Podrobně se problému mizení gradientu věnovaly vědci Hochreiter a Schmidhuber [19], kteří i vyvinuli umělou buňku typu LSTM.

Technika ULMFiT je hlavně založená na použití AWD-LSTM (*ASGD Weight-Dropped LSTM*) principů [41]. Zjednodušený pohled na architekturu AWD-LSTM sítě je uveden na obrázku číslo 3.9, podrobnější popis viz [60]. Jedná se o bezkompromisně nejvyspělejší architekturu pro vyřešení problému zpracování přirozeného jazyka NLP, která ideálně vyhovuje na zpracování a predikci sekvenčních vstupů. Ve svém základu AWD-LSTM sítě kombinují techniky DropConnect a Average Random Gradient Descent [41]. První technika DropConnect vychází z klasické regulační techniky rekurentních neuronových sítí – Dropout, která byla vyvinutá a patentována společností *Google* a její primárním cílem je snížit pravděpodobnost přetrénování sítě [57]. Klasický přístup Dropout spočívá v náhodném resetování podmnožiny aktivací vnitřních neuronů sítě během procesu trénování [57]. Dané vynulování aktivací uzlů v praxi působí jako stimulační signál k zapojení blíž ležících umělých neuronu. Technika DropConnect používaná v AWD-LSTM sítích na rozdíl od vynulování aktivací neuronů provádí náhodné vynulování množiny vah [57].

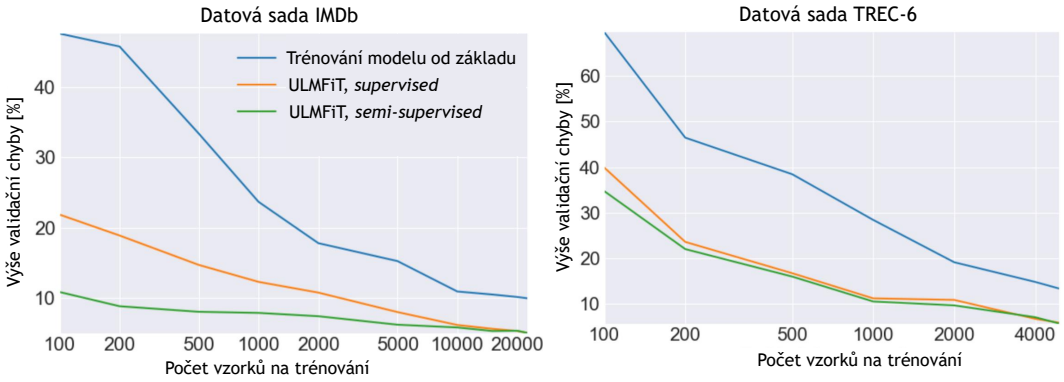


Obr. 3.9: Struktura vrstvy `AWD-LSTM` hlubokých neuronových sítí

Druhá technika ležící v základu `AWD-LSTM` sítí je `Average Random Gradient Descent`. Tato technika optimalizace vychází z optimalizační metody SGD (*Stochastic gradient descent*), která slouží pro optimalizaci procesu nalezení lokálního minima funkce, pomocí algoritmu gradientního sestupu (*gradient descent*). Ve svém základu metoda SGD používá princip, tzv. stochastického přiblížení SAM (*Stochastic Approximation Method*). Metoda ARGD, tj. `Average Random Gradient Descent` používá stejný krok aktualizace gradientu jako i SGD, avšak oproti SGD nevrací váhu spočítanou za poslední iteraci, ale počítá a vrací průměr vah za poslední iterace.

Výhody modelu `ULMFiT`

Hlavní výhodou dané techniky `ULMFiT` je aplikování technik přeneseného učení *transfer learning* na problematiku pochopení přirozeného jazyka (NLP) [62]. Laicky řečeno se jedná o přístup učení hlubokých neuronových sítí ne od samých základů (*from scratch*), ale od určité přetrénované úrovně, viz obrázek číslo 3.10. Použití techniky `ULMFiT` umožní co nejrychlejší konvergenci k finálnímu řešení ze všech existujících v dnešní době přístupů [41]. Dané vlastnosti je dosaženo tím, že základní model neuronové sítě `AWD-LSTM` ležícím v základu ULMFiT, byl přetrénován (*fine-tunned*) a generalizován na obecné datové množině `Wikitext 103`. V podstatě se jedná o takový model naučené umělé sítě, který není zaměřen na vyřešení žádného specifického úkolu a ve své základní variantě nebude mít vynikající finální přesnost při řešení specifických problémů, jako je například filtrování spamových zpráv. V případě, pokud máme za cíl použit výše zmíněný model a dosáhnout *state-of-the-art* výsledků, je nutné generalizovaný model sítě doučit na vyřešení specifického problému, tzv. aplikováním technik přeneseného učení. Silnou stránkou takových řešení je dosažení vysoké přesnosti klasifikace do tříd, při aplikování minimálního počtu epoch



Obr. 3.10: Srovnání `ULMFiT` přeneseného učení oproti klasickému přístupu[15]

trénování, viz obrázek číslo 3.10. Na tomto obrázku je uvedené srovnání zvoleného přístupu trénování `ULMFiT` oproti klasickému trénování od základu (*from scratch*) na veřejně dostupných datových sadách IMDb a TREC-6, viz [41]. Za zmínku také stojí, že v poslední době je daný přístup velice rozšířený, jelikož umožňuje vytvořit robustní model, při minimálním počtu nákladů a velmi efektivně [41]. Hlavní motivaci při vytvoření podobných generalizovaných modelů je možnost distribuovat komplikovaná řešení mezi veřejnost a tím umožnit ještě rychlejší vývoj technického pokroku a zdokonalení odvětví umělé inteligence.

3.3.2 Model `Google BERT`

Charakteristika modelu

Druhá technika, která byla v diplomové práci otestována také spadala do *state-of-the-art* metod sloužících na vyřešení problémů patřících do skupiny NLP (*Natural Language Processing*). Její vznik umožnil analyzovat problematiku zpracování přirozeného jazyka z nového úhlu pohledu. Bezpochyby se jedná o pokročilejší architekturu, která umožnila realizovat dosažení lepších přesností při aplikování na texty se složitou strukturou a velkým množstvím sémantických závislostí mezi slovy [42]. Jak je z nazvu této podkapitoly patrné, jedná se o techniku `BERT` (*Bidirectional Encoder Representations from Transformers*), která byla představena na konci roku 2018 vědeckým týmem[16] společnosti Google pod vedením Jacoba Devlina [43]. Jelikož je úloha klasifikace textů podmnožinou problémů NLP, lze hlubokou síť `BERT` úspěšně

[15]Dané schéma bylo přeloženo z oficiálních statistik dostupných z oficiální dokumentace k metodě hlubokého učení ULMFiT, detailněji viz [41].

[16]Vědecký Google tým byl zastoupen následujícími vědci: Jacob Devlin, Ming-Wei Chang, Kenton Lee a Kristina Toutanova [43].

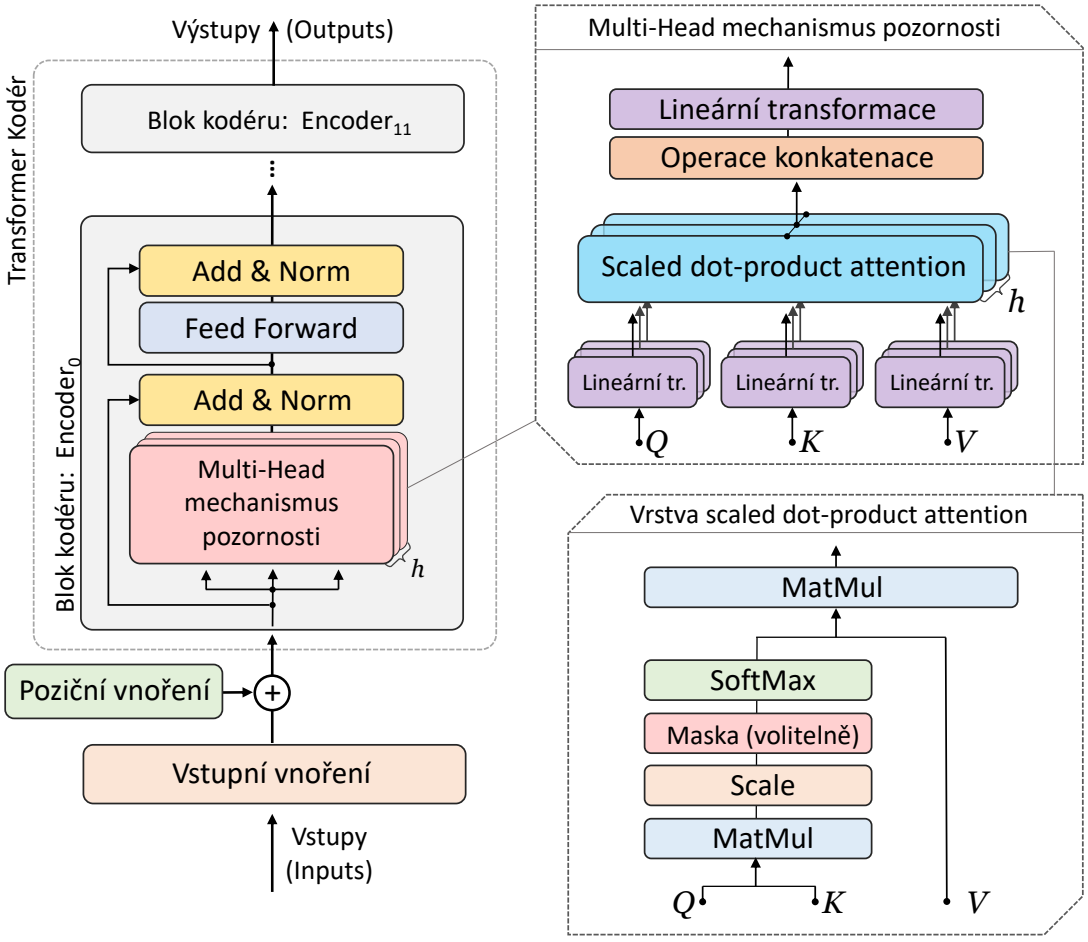
aplikovat i na vyřešení úlohy položené v této diplomové práci. Základní popis techniky `BERT` byl představen v podkapitole číslo 2.4. V rámci této podkapitoly bude princip fungování daného modelu popsán více do hloubky, a to s primární motivací pochopení specifik jeho architektury a klíčových vlastností a bezproblémové aplikace na doménu filtrování nevyžádaného elektronického obsahu.

Model hluboké neuronové sítě `BERT` byl zveřejněn ve stejném roce jako zmiňovaný v předchozí kapitole model ULMFiT. Po svém vzniku, stejně jako ULMFIT, ukázal *state-of-the-art* výsledky ve mnoha odvětvích NLP[17], strukturálně se avšak od `ULMFiT` výrazně lišil. Odlišnosti v architektuře a způsobu zpracování vstupního textu posloužily motivací tento model zprovoznit a otestovat na vyřešení binární klasifikace e-mailové pošty. Sám o sobě se model `BERT` jeví jako obousměrná umělá neuronová síť, tzv. BANN (*Bidirectional Artificial Neural Network*), v jejíž základu leží architektura tzv. transformerů (*transformer architecture*). Oproti sekvenčnímu zpracování vstupních informací, jako je tomu v modelech implementujícím RNN a LSTM sítě, architektura transformerů `BERT` používá přístup založený na pozornosti, tj. *Attention-based approach* [44]. Hlavní novinkou daného přístupu, oproti klasickým způsobům, je odstoupení od rekurentních sítí a přistoupení k mechanismům pozornosti (*attention mechanisms*) sloužících k vytvoření globální propojenosti mezi vstupní a výstupní sekvencí. Navíc, implementace transformerů umožňuje snadněji paralelizovat proces učení neuronové sítě a dosáhnout efektivnějšího zpracování delších textových sekvencí, detailněji viz [47]. Oproti klasickým transformerům `BERT` ve své architektuře neimplementuje dekodéry, ale pouze skupinu kóderů, které jsou sdruženy do dvanácti anebo dvaceti čtyř vrstev [42, 45], viz dále.

Na obrázku číslo 3.11 je zobrazená zjednodušená architektura `BERT` sítě, konkrétně skupina transformer kodérů a *embedding* vrstev. Princip fungování kodéru bude vysvětlen v této podkapitole. Vysvětlení zásad fungování *embedding* vrstev bude částečně uděláno v této podkapitole a následně detailněji spolu s příklady popsáno v části číslo 3.5. Podrobný popis fungování transformerů je možné dočíst z literatury [42, 45, 47]. V první řadě je důležité zmínit, že první kroky zpracování vstupných textových sekvencí a jejich převodu do vice-dimenzionálních vektorů v *attention-based* architektuře jsou podobny klasickému zpracování pomocí RNN sítí, tzn. konceptuálně se blíží k ULMFiT. V první řadě vstupní textová sekvence podléhá tokenizaci, po níž je sekvence rozdělena do dílčích tokenů. Následně dochází k numerikalizaci (*numericalization*) tokenů a jejich převodů na celočíselné hodnoty. Potom pomocí *embedding* vrstev jsou obdržené celočíselné hodnoty převedeny na vícedímenzionální vektory, jejichž dimenzionalita je v rámci trénování definována proměnnou `emb_dim` v modelu `BERT`. Po provedení operaci vnoření slov

[17] Vznik modelu `BERT` posunul hranici GLUE testů na 80,5 % (absolutní přírůst 7,7 %), MultiNLI na 86,7 % (absolutní přírůst 4,6 %) a SQuAD na 93,2 (přírůst byl 1,5 bodů), viz [43].

zahrnující vstupní vnoření (*input embedding*) a poziční vnoření (*positional encoding*) je zakódovaná textová sekvence zastoupena maticí $X = Z + P$, která je tvořena součtem matic Z (vicedímenzionální tokeny po vnoření) a P (kódované pozice tokenů), detailněji viz podkapitolu číslo 3.5. Velikost matice X je tedy definována jako `input_length` × `emb_dim`, kde `input_length` – maximální délka vstupní sekvence a je pro síť `BERT` omezena na 512 tokenů, a to včetně pomocných značek. V dalších



Obr. 3.11: Zjednodušená architektura kodéru transformeru `BERT`[18]

krocích dochází k zpracování výstupní matici X řadou kodérů, které se uspořádávají vrstevně za sebou a tvoří jádro hluboké neuronové sítě. V modelu `BERT` je struktura kodérů převzatá z klasických transformer sítí, viz [43, 47]. V závislosti na typu `BERT` sítě, může dané jádro obsahovat od 12 až do 24 bloků kodéru, viz charakteristiku existujících modelů dole. Úlohou každého dílčího bloku je odhalení vnitřních závislostí numerikalizovaných a zakódovaných tokenů a předání určeného odhadu na vstup do další vrstvy [45]. Jednodušeji řečeno, daný přístup umožňuje neuronové sítě zachytit

[18]Podrobnější popis principů fungování transformerů je možné dohledat ve zdrojích [42, 45, 47].

komplexnější závislosti mezi slovy a přesněji zakódovat závislosti vstupní sekvenci. Prvním blokem kodéru je blok *multi-head attention* neboli vícevrstevný mechanismus pozornosti, který se skládá z h hlaviček pozornosti[19]. Pomocí vnitřní funkce *scaled dot-product attention* (viz obrátek číslo 3.11) je matice *attention* aplikováním různých váhových matic W (*weight matrices*) vypočítaná h-krát a následně, pomocí konkatenace (*concatenate*) a lineárních transformací převedena na jednu výstupní matici. Výsledku každého paralelního výpočtu se říká *head*. Propojenost dílčí váhové matici W s příslušným *head* prvkem se bude v následujícím vzorci značit indexem i. Celkově lze funkci *MultiHead* matematicky popsat následujícím způsobem [43, 47]:

$$\text{MultiHead}(Q, K, V) = \text{Concatenate}(head_1, head_2, \ldots, head_h) \cdot W^O,$$

kde je prvek $head_i$ možné vypočítat aplikováním vzorce:

$$head_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V).$$

Vstupní matice do funkce *MultiHead* se značí písmeny Q, K a V (viz obrázek 3.11). Horními indexy Q, K, V se u váhových matic W_i^Q, W_i^K a W_i^V značí příslušné váhové matice, které jsou aplikovány na konkrétní vstupní tensor. Přičemž pro váhové matice platí: $W_i^K, W_i^Q \in \mathbb{R}^{d_{model} \times d_k}, W_i^V \in \mathbb{R}^{d_{model} \times d_v}$. Matice $W^O \in \mathbb{R}^{hd_v \times d_{model}}$ je finální váhová matice, která je aplikována po provedení operace konkatenace [47]. Hodnota $d_{model} =$ `emb_dim` je dimenzionalita modelu hluboké neuronové sítě a hodnoty $d_k = d_v = d_{model}/h =$ `emb_dim`$/h$, detailněji viz [42, 47]. Pokud se detailněji zaměříme na funkci *scaled dot-product attention* (viz obrátek číslo 3.11), zjistíme, že jednotlivé vstupní matice Q, K a V jsou zpracovány následujícím způsobem[20]:

$$\text{Attention}(Q, K, V) = \text{Softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right) \cdot V.$$

Přičemž z obrázku je patrné, že aplikování masky po operaci *Scale* je u transformerů volitelné a konkrétně v části `BERT` kodéru může být opuštěno [43]. Po průchodu skrz blok *multi-head attention* vstupuje matice (označíme jako x) do bloku *Add & Norm*, který lze možné matematicky popsat:

$$\text{LayerNorm}(x + \text{Dropout}(\text{Sublayer}(x))).$$

Hodnota *dropout* u dané funkce je nastavena na 10 %. Zjednodušeně řečeno, v dané etapě dochází k přečtení původní vícedimenzionální reprezentaci vstupního tokenu

[19] Množství hlaviček se liší od vybraného `BERT` modelu a může se pohybovat od 12 do 16 *attention heads* prvků, viz popis `BERT` modelů v textu anebo zdroje [42, 43, 45].

[20] Pro zjednodušení a zpřehlednění vzorce byla operace násobení váhovými maticemi W_i^Q, W_i^K a W_i^V v daném kroku opuštěna. Celý popis fungování kodéru transformeru lze zjistit z [47].

k reprezentaci jeho závislostí s jinými tokeny. V důsledku čehož je původní reprezentace sémanticky obohacena o blíž ležící slova vstupní textové sekvence. V posledním kroku u každého kodéru dochází k zpracování matic pomocí plně propojených vrstev dopředné neuronové sítě FFN (*Feef-Forward Network*), viz obrázek 3.11. Síť FFN je na každou vstupní pozici aplikována zvlášť a za stejných podmínek [47]. Data prochází dvěma lineárními transformacemi, mezi nimiž je použitá aktivační funkci typu ReLU, matematicky lze operace popsat [47]:

$$\mathrm{FNN}(x) = \max(0, xW_1 + b_1)W_2 + b_2.$$

Na konci bloku probíhá operace *Add & Norm*, která je skoro identická k popsané v předchozím textu, s jedinou odlišnosti, že zpracovává vstupní matice před a po aplikování FNN, detailněji viz [42, 43, 47].

Model `BERT` je předtrénován technikou učení bez učitele (*unsupervised learning*) ne na vytvoření jazykového modelu [45], ale na vyřešení úloh maskovaného jazykového modelování MLM[21] (*Masked Language Modeling*) a predikování další věty na základně definovaného vstupu NSP (*Next Sentence Prediction*) [44]. Daná vlastnost umožňuje aplikování technik přeneseného učení na vyřešení úloh klasifikace sentimentů (*sentiment classification*), zodpovězení na otázky (*question answering*) a dalších [44]. Následujícím odlišujícím specifikem `BERT` je odstoupení od klasického zpracování textové sekvence z jedné strany, tzn. zleva doprava (*left-to-right*) anebo zprava doleva (*right-to-left*) a provedení dvousměrné analýzy (*bidirectional analysis*) na všech svých vrstvách [43]. Dané vylepšení poskytlo možnost propojení tokenizovaných vstupních slov na obě strany sekvence, tj. na předchozí a budoucí token současně. Aplikování hlubokých obousměrných transformerů DBT (*Deep Bidirectional Transformers*) umožnilo vytvořit přesnější mapu sémantických vazeb vstupní sekvence, a tím dosáhnout vyšší finální přesnosti natrénovaných modelů, viz [43].

`BERT` je integrován do frameworků `PyTorch` a `TensorFlow`. Existují různé variace `BERT` architektur, jejichž volba se liší od zvolené NLP úlohy, požadavků prostředí, nároků na výstupní model atd. Primárně lze dostupné modely `BERT` rozdělit na dvě skupiny, a to na základě celkového množství vrstev, tj. `BERT-Base` s 12 vrstvami a `BERT-Large` obsahující 24 vrstvy. Výčet nejpoužívanějších z nich je uveden níže:

- `BERT-Base (Uncased)`: 12 vrstev, 768 skrytých prvků, 12 *attention heads* prvků, 110 milionů optimalizačních parametrů, podpora angličtiny;
- `BERT-Base (Cased)`: 12 vrstev, 768 skrytých prvků, 12 *attention heads* prvků, 110 milionů optimalizačních parametrů, podpora angličtiny;

[21] Úloha MLM (*Masked Language Modeling*) spočívá ve predikování slova zastíněného maskou na základě kontextu zjištěného z vedlejších tokenů ve větě.

- `BERT-Base (Chinese)`: 12 vrstev, 768 skrytých prvků, 12 *attention heads* prvků, 110 milionů optimalizačních parametrů, současná podpora zjednodušené a tradiční čínštiny;
- `BERT-Base (Multilingual Cased)`: 12 vrstev, 768 skrytých prvků, 12 *attention heads* prvků, 110 milionů parametrů, 104 podporované jazyky;
- `BERT-Large (Uncased)`: 24 vrstvy, 1024 skrytých prvků, 16 *attention heads* prvků, 340 milionů optimalizačních parametrů, podpora angličtiny;
- `BERT-Large (Cased)`: 24 vrstvy, 1024 skrytých prvků, 16 *attention heads* prvků, 340 milionů optimalizačních parametrů, podpora angličtiny.

Jak je vidět z výčtu dostupných modelů, jedná se o multijazykovou neuronovou síť, kterou lze aplikovat na většinu jazyků včetně angličtiny a češtiny (ve verzi `Multilingual Cased`) [48]. Za zmínku také stojí, že model `BERT-Base` byl autory vyvinut s motivací provedení přímého srovnání s modelem `GPT` od OpenAI, který svými velikostními parametry se k `BERT-Base` velmi blíží [43], avšak oproti technikám používaných v `BERT` provádí analýzu tokenů pouze zleva doprava, je natrénován na jiné datové sadě a rychlost její učení se dynamicky neodvíjí od druhu úlohy [43, 49].

Navíc, oproti klasickým modelem implementujícím mechanismy pozornosti, nepoužívá `BERT` ploské struktury pozornosti (*flat attention structure*). V závislosti na zvolené verzi implementuje 12 anebo 24 *attention* vrstev, které navíc zahrnují 12 anebo 16 tzv. *self-attention heads* prvků. Nastavované váhy sítě během trenování nejsou distribuovány mezi jednotlivými vrstvy, a proto ve své maximální verzi `BERT-Large` realizuje až do $16 \cdot 24 = 384$ samostatných mechanismů pozornosti.

Výhody modelu `Google BERT`

Stejně jako `ULMFiT`, síť `BERT` má dvě základní výhody spočívající v její generalizaci a předtrénovanosti na rozsáhlé datové sadě. Generalizace umožňuje dotrénovat existující síť na vyřešení konkrétní NLP úlohy. Daná vlastnost je docílena tím, že je síť defaultně schopná pracovat s různorodými texty a odhalovat základní sémantické vazby pomocí mechanismů pozornosti. Druhá vlastnost techniky `BERT`, tj. její předtrénovanost souvisí s možností aplikování technik přeneseného učení. Předtrénování modelu bylo docíleno aplikováním techniky učení bez učitele na vyřešení úloh MLM a NSP. Při provedení učení sítě a adaptaci na konkrétní problém není nutné ji trénovat od základu. Pro přizpůsobení modelu na zvolenou úlohu je tedy zapotřebí dotrénovat pouze výstupní vrstvy neuronové sítě [43]. Předtrénovaný model při své inicializaci již obsahuje vytvořené vnoření slov. Slovník modelu je zastoupen 30 000 tokeny[22]. Předtrenování sítě bylo provedeno na dvou datových sadách `BooksCorpus` (800 milionů slov) a `English Wikipedia` (2 500 milionů slov) [43].

[22] `BERT` realizuje techniku WordPiece pro vytvoření vnoření slov, detailněji viz z [43].

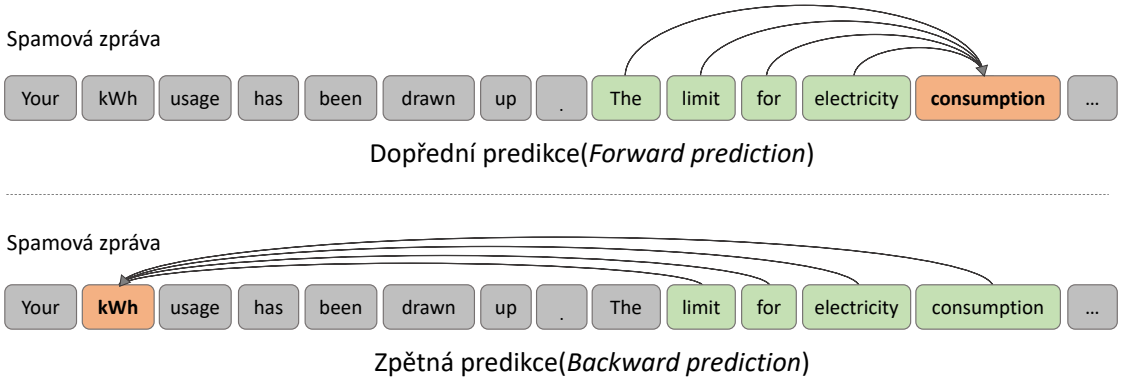
3.3.3 Model `Google Brain XLNet`

Charakteristika modelu

Třetím modelem, který bylo zvoleno otestovat v rámci vytvoření klasifikátoru nevyžádané pošty byl `XLNet` (*Generalized Autoregressive Pretraining Model*). Model této neuronové sítě byl zvolen z takového důvodu, že se v řádě svých klíčových vlastností velmi blíží k detailně popsánému, v rámci kapitoly číslo 3.3.2, modelu `BERT`. Avšak se nejedná o pouhou kopii sítě `BERT`, ale o samostatný druh architektury hluboké neuronové sítě, jehož primárním stavebním prvkem, stejně jako v případě sítě `BERT`, jsou transformery [51]. Dá se říci, že v současné době je architektura transformerů nejvíce slibující pro vyřešení úloh patřící do NLP. Samá o sobě, síť typu `XLNet` vznikla v červnu roku 2019, tj. později, než zmiňovaná síť `BERT` a ULMFiT [51, 52]. Vývoj, návrh architektury a předtrénování sítě provádělo vědecké oddělení Google Brain, které se primárně zabývá odvětvím umělé inteligence a strojového učení [53]. Hned po svém představení síť `XLNet` překročila síť `BERT`[23] ve dvaceti NLP úlohách a dosáhla *state-of-the-art* výsledků v osmnácti z nich, do nichž je možné zařadit úlohy zodpovězení na otázky (*question answering*), inference přirozeného jazyka (*natural language interence*), analýzu sentimentů (*sentiment analysis*) a hodnocení dokumentů (*document ranking*) [51, 52, 53]. Síť je stejně jako předchozí modely schopna řešit úlohy textové klasifikace, a to jak její binární podobu, tak i multiznačkovou [55].

Primárně `XLNet` vychází z existujícího modelu `Transformer-XL`, přičemž si klade za cíl vylepšit jeho architekturu a umožnit efektivnější zpracování delších textových sekvencí [51]. Jak bylo zmíněno výše, daná síť patří do skupiny algoritmů typu ARLM (*Autoregressive Language Model*). Jiným populárním představitelem této skupiny je model hluboké neuronové sítě `GPT` a také jeho nástupce model `GPT-2` [53]. Charakterním příznakem daných modelů je vlastnost, že umí na základě kontextu, tj. blíž ležících slov, předpovídat následující slovo [53]. Hlavní výhodou takového přístupu je to, že naučené modely umí velmi dobře řešit úlohy na generování textu, jelikož je generování textu přímo založeno na doplnění nejlíp lexikálně a významově vyhovujícího slova na základě předchozí textové sekvence [53, 54]. Algoritmy typu ARLM je možné rozdělit do dvou základních skupin. První skupinu tvoří algoritmy, které pro generování textu používají dopřední předpověď, tj. *forward prediction*. Druhá skupina algoritmů je založená na aplikování zpětné předpovědi, tj. *backward prediction*. Za zmínku také stojí skutečnost, že ve své klasické podobě neumí modely typu ARLM předpovídat do obou stran zároveň, co je samozřejmé možné považovat za jakousi nevýhodu [51, 53]. Dopředu lze ale říci, že se síť `XLNet` během svého trénování snaží výše zmíněnou nevýhodu eliminovat, viz dále. Na obrázku číslo 3.12 je vidět základní princip fungování AR modelů [53].

[23] Jedná se o modifikaci sítě `BERT` ve své maximálně dostupné verzi, tj. `BERT-Large (Cased)`.



Obr. 3.12: Princip fungování dopředných a zpětných *autoregressive* modelů

V prvním případě probíhá dopřední odhadování slova na základě kontextu zjištěného z předchozích slov[24] (*forward prediction*). V druhém příkladu je jednotka energie předpovězena na základě kontextu zjištěného z následujících slov, tj. aplikováním principu zpětné predikce (*backward prediction*). Oproti tomu, z informací dostupných v podkapitole číslo 3.3.2, je patrné, že není síť `BERT` založena na *autoregressive* modelu, ale stojí na použití jazykového modelu auto-kodéru (*autoencoder language model*) [53]. Jeho hlavní výhodou je obousměrná možnost analýzy kontextu, a to z důvodu lišicího se principu fungování. Principiální odlišností je nahrazení chybějících tokenů speciální maskou (značka `[MASK]`), a aplikování obousměrné techniky analýzy k její odhalení, detailněji viz 3.3.2. Nevýhoda techniky používané v `BERT` spočívá v tom, že maskování tokenů probíhá pouze v rámci předtrenování modelu, nikoliv během přeneseného učení [53, 54]. Navíc, se předpokládá, že predikované tokeny jsou na sobě nezávislé, což ve finále může negativně ovlivnit úspěšnost odhalení sémantických vazeb při přeneseném učení modelu [54].

Technika `XLNet` si bere za cíl eliminovat nevýhody klasických *autoregressive* modelů a zároveň převzít obousměrné vlastnosti sítě `BERT` [51]. Z takového důvodu se používá permutační technika PLM (*Permutation Language Modelling*) během fáze předtrenování. Jak vyplývá z její nazvu, při provedení optimalizace parametrů sítě dochází k vyzkoušení různých permutací[25] vstupní tokenizované textové sekvence [53]. Přesněji řečeno síť `XLNet` neprovádí pouze dopřední a zpětnou predikci, ale realizuje její různé permutace [54]. Samozřejmě, pokud se na danou úlohu podíváme ze statistického úhlu pohledu, pro libovolnou n-prvkovou množinu celkově existuje $n!$ permutací [53, 54]. V praxi to znamená, že v případě jednoho

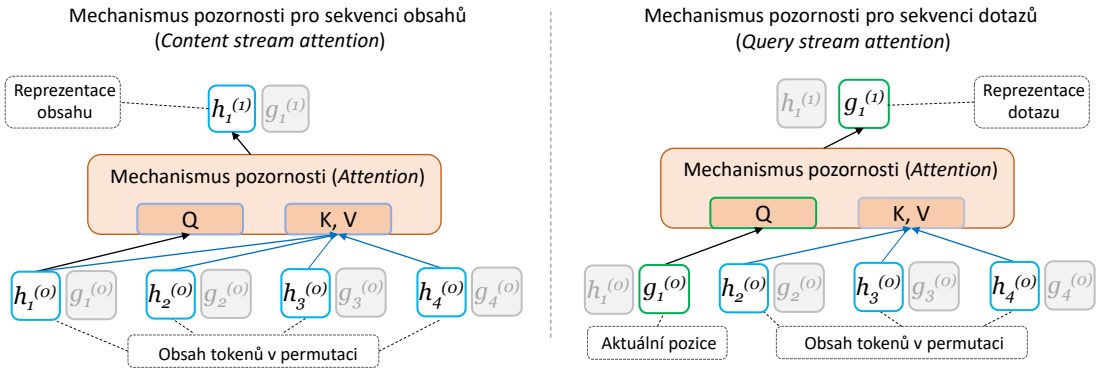
[24]Daný princip se hojně používá v aplikacích, které pomáhají psát textové zprávy a nabízejí na doplnění nejvhodnější slova.

[25]Pojem permutace v matematice a statistice zastupuje jedno z možných uspořádání n-prvkové množiny, a to takovým způsobem, aby výsledná množina obsahovala každý prvek pouze jednou.

tokenizovaného vstupu s maximální délkou 512 tokenů, by síť měla zanalyzovat $512! = 3,4773 \times 10^{1166}$ možností, což není v logaritmickém čase spočitatelné, ani pro jednou textovou sekvenci. Z takového důvodu síť `XLNet` v závislosti na nastaveních generuje pouze několik permutací, které v budoucnu používá pro své výpočty. Matematicky se při řešení permutačního jazykového modelování snažíme dosáhnout stavu [51, 54]:

$$\max_{\theta} \mathbb{E}_{\mathrm{Z} \sim \mathcal{Z}_T} \left[\sum_{t=0}^{T} \log p_\theta(x_{z_t} | x_{\mathrm{Z}<t}) \right].$$

Kde Z_T je množina všech permutací z. Faktorizační řad se označuje jako Z. Délka vstupní tokenizované sekvence se značí jako T. Symbolem x_{z_t} se značí t-tý token permutace z. Přičemž $x_{\mathrm{Z}<t}$ je libovolný token ze stejné permutace z, ale ležící před zmíněným dříve prvkem x_{z_t}. Navíc, v matematickém vyjádření permutačního jazykového modelování vystupuje pravděpodobnostní funkce (*likelihood function*), která se značí jako p_θ a používá se pro vyhodnocení úspěšnosti statistického modelu. Použití permutací umožňuje dosáhnout takového stavu, kde bude zvolený token x_{z_t} předpovězen z různých pozic v tokenizované sekvenci, a to z takového důvodu, že každý z tokenů v permutované sekvenci z splňující podmínku $x_{\mathrm{Z}<t}$ bude použit během předpovědi x_{z_t} tokenu. Aplikováním daného fíglu dojde k vyřešení omezení daných charakterními vlastnostmi AR (*Autoregressive*) modelů. Na druhou stranu práce s permutacemi původní množiny přináší problémy, které klasické transformer sítě řešit neumí [54]. Hlavním z nich je neschopnost transformerů rozdělit informaci o aktuální pozici tokenu s vnořením (*embedding*) jeho kontextu [51, 54]. Síť `BERT` takovým problémem netrpí, jelikož nepracuje s permutacemi. `XLNet` pro eliminaci této negativní vlastnosti aplikuje dva mechanizmy pozornosti zobrazené na obrázku 3.13 [51].

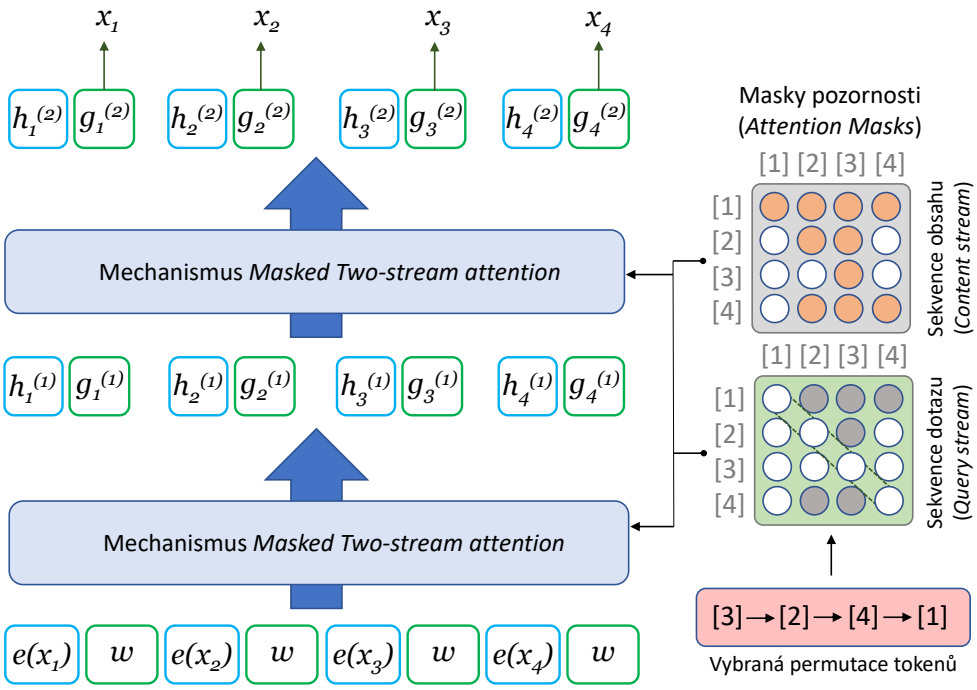


Obr. 3.13: Mechanismy pozornosti používané v síti `XLNet`

Prvním z nich je mechanismus pozornosti pro sekvenci obsahů CSA (*Content Stream Attention*), který je převzat ze standardního transformer mechanismu *self-attention* popsaným v podkapitole 3.3.2 [51]. Druhým je mechanismus pozornosti pro sekvenci dotazů QSA (*Query Stream Attention*), který tvoří alternativu k použití maskování textu v `BERT` [51]. Na obrázku číslo 3.13 je zobrazena zjednodušená podoba funkcí CSA a QSA, která pracuje pouze s čtyřmi tokeny. Každý z nich je v `XLNet` modelu vyjádřen dvojicí $h_i^{(j)}$ zastupující obsah tokenu a $g_i^{(j)}$ vyjádřující jeho aktuální pozici [54], viz obrázek 3.13. Primárním účelem *Content Stream Attention* je provedení reprezentaci obsahu na základě obsahu zjištěného z jiných tokenů. Z obrázku je patrné, že je zde použitá pouze množina $\{h_q^{(j)}\}$ a pro reprezentaci obsahu tokenu $h_1^{(1)}$ se používají všechny obsahy tokenů v sekvenci, tj. $h_1^{(0)}$ až $h_4^{(0)}$. Množina prvků $g_i^{(j)}$ obsahující aktuální reprezentace dotazů (*query representation*) tokenů není ve funkci použitá [54]. Navíc, je z obrázku patrné, že v prvním případě pro proměnnou Q platí: $Q = h_1^{(0)}$ a pro proměnné $K, V = \{h_1^{(0)}, h_2^{(0)}, h_3^{(0)}, h_4^{(0)}\}$, detailněji viz [51]. Druhou funkcí na obrázku číslo 3.13 je QSA, neboli *Query Stream Attention*. Daný mechanismum pozornosti pro predikci dotazů nesmí brát v potaz obsah hledaného tokenu, tj. v našem případě je to $h_1^{(0)}$, a proto pro predikci *query* jsou zde použity hodnoty zbylých tokenů $h_2^{(0)}$ až $h_4^{(0)}$ a také aktuální pozice predikovaného tokenu $g_1^{(0)}$. Proměnná Q je v daném případě zastoupená: $Q = g_1^{(0)}$ a hodnoty $K, V = \{h_1^{(0)}, h_2^{(0)}, h_3^{(0)}\}$.

Na obrázku číslo 3.14 [51] jsou v grafické podobě uvedeny všechny výpočty, které se provádí v hluboké sítě v rámci trénování *permutation language modeling* s použitím mechanismu maskované dvoukanálové pozornosti (*Masked two-stream attention*), znázorněnou a popsanou v textu předtím.

Popis obrázku 3.14 bude za účelem lepšího pochopení proveden odspodu. V první řadě jsou hodnoty $h_i^{(0)}$ a $g_i^{(0)}$ spočítány jako $h_i^{(0)} = e(x_i)$ a $g_i^{(0)} = w$, detailněji viz oficiální článek [51]. Následně po inicializaci spodní vrstvy probíhá zpracování informací dvěma *attention* mechanismy, které postupně, tj. aplikováním dvou masek na tokenizovaná data, provádí kódování vstupní textové sekvence [54]. Pokud se zaměříme na pravou část obrázku [51], tak zjistíme, že `XLNet` pro své výpočty používá dvě masky pozornosti (jednu pro každý) [54]. V případě, jestli jsou na vstupu neuronové sítě čtyři tokeny uspořádané do posloupnosti $\{[1] \rightarrow [2] \rightarrow [3] \rightarrow [4]\}$, jejichž náhodně zvolená permutace je, například, $\{[3] \rightarrow [2] \rightarrow [4] \rightarrow [1]\}$ (v souladu s prezentovaném obrázkem 3.14), potom budou masky pozornosti zastoupeny maticemi velikostí 4×4 [51]. V případě matici sloužící pro mechanismus *content stream attention* je její hlavní diagonála zabarvená, což znamená, že při provedení výpočtů vidí každý z tokenů na svůj obsah (daná závislost je také vidět v levé části obrázku 3.13). Následně, pokud se detailněji zaměříme na vybranou permutaci, tak zjistíme, že pro zjištění kontextu tokenu [1] je zapotřebí zahrnout do masky kontext posloupnosti před nim $\{[3] \rightarrow [2] \rightarrow [4]\}$ a samotný token [1] (zabarvený první řádek matice).



Obr. 3.14: Proces *permutation language modelling* během předtrénování `XLNet`

Pro token [2] musí být do masky zahrnuté dva tokeny, konkrétně {[3] → [2]}. Jelikož před tokenem [3] neleží žádný jiný, bude v masce zastoupen pouze jedním zabarveným prvkem na pozici [3, 3]. Čtvrtý token [4] je v matici zastoupen sekvenci {[3] → [2] → [4]}. Druhá matice používaná pro *query stream* má prázdnou hlavní diagonálu, protože v daném případě jednotlivé prvky nesmí nahlížet do vlastních kontextů [51, 54]. Principiálně avšak obarvení probíhá obdobným způsobem.

V současné době existují dvě variace sítě `XLNet`, které jsou integrované do knihovny `PyTorch`. Zprostředkovávají se přes balík `Transformers`, který je možné nainstalovat pomocí nástroje `pip`. Níže je uveden stručný popis klíčových vlastností modelů [52].

- `XLNet-Based (Cased)`: 12 vrstev, 768 skrytých prvků, 12 *attention heads* prvků, 110 milionů optimalizačních parametrů, podpora angličtiny;
- `XLNet-Large (Cased)`: 24 vrstvy, 1024 skrytých prvků, 16 *attention heads* prvků, 340 milionů optimalizačních parametrů, podpora angličtiny;

Pro oba modely platí, že byly natrénovány na datových sadách jako BookCorpus, English Wikipedia, Giga5, ClueWeb 2012-B, Common Crawl [51]. Po provedení filtrování a tokenizace výše zmíněných datových sad, a to aplikováním technik `SentencePiece` bylo ve finále obdrženo 32,89 B tokenizovaných dat, viz oficiální dokument [51]. Velikostně se modely blíží k modelům `BERT-Large` a `BERT-Base`. Maximální délka tokenizované sekvence pro oba modely činí 512. Minimální požadavky na zprovoznění modelu `XLNet-Large` jsou 16 GB GPU paměti, z takového

důvodu v rámci této diplomové práce bude při testování použita verze `XLNet-Based`. Za zmínku také stojí skutečnost, že jsou v současné době modely `XLNet` monolingvní, tzn. podporují pouze angličtinu [53]. Níže budou stručně popsány další výhody modelu `XLNet` .

Výhody modelu `Google Brain XLNet`

Výhody modelu `XLNet` se primárně odvíjí od silných stránek generalizovaných předtrénovaných jazykových modelů [51]. A stejně jako u modelů `ULMFiT` a `BERT` spočívají v možnosti aplikování na různé úlohy ze skupiny NLP, konkrétně přidáním odpovídajících výstupních vrstev a také ve vlastnosti umožňující provádět učení sítě `XLNet` ne od samých základů [51]. Podrobně jsou daná specifika rozepsána v podkapitolách číslo 3.3.1 a také v 3.3.2. Jak bylo zmíněno na začátku této podkapitoly výběr na model `XLNet` padl z důvodu ukázání lepších výsledků oproti `ULMFiT` a `BERT` [52], dobré dokumentace a integrace do knihovny `PyTorch`, co ve finále umožní co nejrychleji začíst s napojením modelu na datovou sadu vybraných e-mailových zpráv a aplikovat techniky přeneseného učení na dostupném hardware. Do dalších zřetelných výhod `XLNet` je možné zařadit použití sofistikovaných způsobů zpracování textových sekvencí pomocí permutačních technik a také dobrou úroveň předtrenování na kvalitní a velké datové sadě [53, 54]. Pokud přímo srovnáme modely `BERT` a `XLNet`, tak zjistíme, že `XLNet-Large`[26] byl předtrénován na rozsáhlejší datově sadě obsahující 113 GB textových dat a dohromady 33 miliardy slov [51, 56]. Oproti tomu, byl `BERT` přetrénován na 16 GB surových dat, jejichž finální slovní zásoba se rovnala 3,3 miliardě slov [56]. Na předtrenování `XLNet` bylo použito 512 hardwarových TPU v3 jednotek, které prováděly trénování sítě 5,5 dnů [51]. Pro lepší přehlednost v tabulce číslo 3.5 bylo vytvořeno srovnání chybovosti[27] modelů `BERT`, `ULMFiT` a `XLNet` při provedení textové klasifikace na různých datových sadách.

3.4 Proces učení modelu `ULMFiT`

Realizovat výše zmíněnou techniku `ULMFiT` bylo možné použitím speciální knihovny `fast.ai`, která se zaměřuje na vývoj, trénovaní a vyhodnocení neuronových sítí [62]. Do hlavních výhod knihovny `fast.ai` je možné zařadit poskytování vestavěných (*build-in*) nástrojů, které umožňují mnohem efektivněji docílit požadováného výsledku a dosáhnout co nejvyšší přesnosti během provedení klasifikace. Konkrétně, knihovna `fast.ai` umožňuje automatizovat předpřípravu datové sady a napojit ji

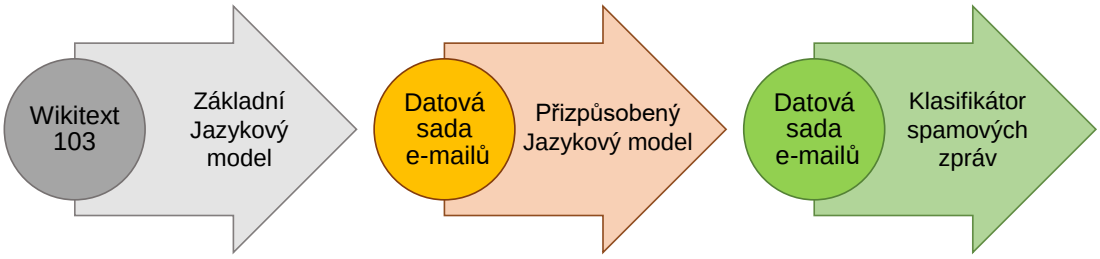
[26] Z oficiálních statistik bylo zjištěno, že model `XLNet-Base` byl naučen na stejném datovém korpusu jako `BERT`, tj. obsahujícím 16 GB dat.

[27] Tabulka chybovosti, neboli *error rates*, byla částečně převzata a přeložena z [51]. Za zmínku stojí, že čím je hodnota chybovosti menší, tím je lepší výsledek modelu sítě.

Tab. 3.5: Srovnání chybovosti modelů `XLNet`, `ULMFiT` a `BERT`

Model sítě	`IMDB`	`Yelp-2`	`Yelp-5`	`AG`	`Amazon-2`	`Amazon-5`
`CNN sítě`	–	2,90	32,39	6,57	3,79	36,24
`ULMFiT`	4,60	2,16	29,98	5,01	–	–
`BERT-Large`	4,51	1,89	29,32	–	2,63	34,17
`XLNet-Large`	**3,20**	**1,37**	**27,05**	**4,45**	**2,11**	**31,67**

na vstup do první vrstvy neuronové sítě, tzn. provést operace *Tokenization*, *Numericalization*, detailněji viz kapitola číslo 2. Následně, knihovna `fast.ai` zjednodušuje a urychluje proces vytvoření jazykového modelu i umožňuje postavit nad nim klasifikátor, který je orientován na vyřešení konkrétní úlohy, viz obrázek číslo 3.15. V samotném základu knihovny `fast.ai` leží `Python` knihovna `PyTorch` založená na knihovně `Torch` [62]. Tyto dvě knihovny umožňují pracovat s modely hlubokých neuronových sítí a provádět výpočty nad tenzory. Zavedení datových struktur – tenzorů dovoluje realizovat mnohonásobné urychlení procesů výpočtů tím, že jsou vykonávány přímo na GPU[28] (*Graphics Processing Unit*) [59]. Zjednodušeně proces učení vybrané neuronové sítě pomocí metody `ULMFiT` je možné popsat pomocí obrázku číslo 3.15. Z obrázku je patrné, že celý proces vytvoření klasifikátoru spamových zpráv je možné rozdělit do třech hlavních etap. [41, 62] V prvním kroku dochází



Obr. 3.15: Základní pohled na učení pomocí ULMFiT

k natrénovaní generalizovaného jazykového modelu neuronové sítě `AWD-LSTM` na základě obecného slovníku obsahujícího nesrovnatelně velkou datovou sadu klíčových tokenů [41, 65, 59]. Při implementaci tohoto kroku se použila datová sada `Wikitext 103`, která bude detailněji popsána níže. V druhém kroku je primární důraz kladen na

[28]Při spouštění kódu na GPU dochází k zavedení paralelizace výpočtů s cílem snížení času nutného na vykonání programu. V závislosti na druhu úlohy, paralelizace umožňuje dosáhnout až stonásobného zrychlení oproti vykonání programů na klasickém CPU. Samozřejmě do nevýhod paralelizace je možné zařadit vyšší složitost kódu a také vykonání pouze omezeného množství úloh.

přizpůsobení vytvořeného v prvním kroku generalizovaného jazykového modelu na řešení konkretního problému. Konkrétně se jedná o provedení procesu přeneseného učení *transfer learning*, ale při použití datové sady, která byla předpřipravena v rámci podkapitoly číslo 3. Během této etapy dochází k provedení finálního předzpracování vytvořené datové sady a inicializaci první vrstvy neuronové sítě `AWD-LSTM` [41]. V rámci třetího kroku dojde k použití vytvořeného jazykového modelu pro vyřešení klasifikační úlohy. V rámci této části bude použita testovací a validační datové sady obsahující označkované *labeled* e-mailové zprávy. Za zmínku také stojí skutečnost, že při provedení prvních dvou kroku se trénování sítě `AWD-LSTM` probíhá bez učitele, tzn. že se aplikují techniky *unsupervised-learning*. V rámci finální etapy učení probíhá pod dohledem a neuronová síť se na základě přidaných značek učí odhadovat původ e-mailových zpráv. Hlavní výhodou aplikování takového přístupu je dosažení *state-of-the-art* přesnosti klasifikace textů, vytvoření velmi robustního a inteligentního systému schopného úspěšně odolat dynamickou změnu spamových technik.

3.4.1 Přizpůsobení datové sady k modelování

V rámci daného kroku probíhá finální předzpracování datové sady e-mailů s cílem vytvoření textového shluku *Text Data Bunch* na základě kterého bude v budoucnu možné udělat jazykový model k rozpoznání spamu [63]. V případě pokud bychom aplikovali modely neuronových sítí na problém klasifikace spamových obrázků, proces napojení obrazů by byl mnohem jednoduší, a to z takového důvodu, že je obraz tvořen množinou pixelů, kde každý pixel může nabývat RGB hodnot v rozsahu od 0 do 255 [59]. V případě textových dat se jedná o množinu slov a existujících lingvistických vazeb mezi nimi (gramatická pravidla, lexikální významy apod.), které neumožňují aplikování klasických matematických funkcí [65, 62, 59]. Proto hlavním cílem finálního přizpůsobení dat je převod těchto slov a vazeb mezi nimi na čísla se zachováním co největšího množství podstatných informací [58].

Z obrázku je patrné, že v rámci prvního kroku dochází k načtení a předzpracování datových sad e-mailů. Po nahrání jsou data rozdělena do třech množin, tzv. *DataFrame*, pro účely trénování, validace a testování. Navíc je každé e-mailové entity v datových sadách přiřazena značka, tzv. *label*, detailněji viz podkapitola číslo 3.2.5. Jak bylo uvedeno výše, v případě vytvoření jazykového modelu e-mailů se jedná o provedení učení neuronové sítě bez učitele, jehož cílem je rozšíření obecného jazykového modelu `Wikitext 103` na takovou slovní zásobu, která se vyskytuje ve spamových zprávách [62, 65]. Z takového důvodu jsou přiřazené značky *label* specifikující zda se jedná o spam anebo ham, jsou v rámci prvního kroku ignorovány.

K urychlení etapy přizpůsobení dat k modelování byla použita výše zmíněna knihovna `fast.ai`. V rámci tohoto kroku došlo k provedení procesů *Tokenization*

```
# Vytvoření objektu obsahujícího předzpracované datové sady e-mailů. Hodnota @seed
# je vložena s účelem znáhodnění procesu modelování datových sad
ulmfitDataProcessing = self.UlmfitDataProcessing(spam_dataset, ham_dataset, seed=99);

# Naplnění konkrétních datových sad pro následující zpracování
train_frame = ulmfitDataProcessing.get_trainframe()
validation_frame = ulmfitDataProcessing.get_validationframe()
test_frame = ulmfitDataProcessing.get_testframe()

# Vytvoření instance @data_language_model obsahující předzpracované
# datové sady @train_frame a @validation_frame
data_language_model = TextLMDataBunch.from_df('./', train_df=train_frame,
valid_df=validation_frame)
```

Obr. 3.16: Výpis kódu sloužícího k přizpůsobení datové sady k modelování

a *Numericalization* [41]. Proces převedení existujících v původní datové sadě slov na tokeny, tj. proces *Tokenization*, spočívá v přiřazení jednotlivých slovům speciálních značek – tokenů. Při provedení značkování textu dochází k rozdělení vět na menší části, které zastupují buď jednotlivá slova, fráze anebo symboly. Rozdělení textu na tokeny neprobíhá pouze na základě vyskytujících se v textu mezer, ale v potaz jsou také brána interpunkční znaménka, které umožňuji líp pochopit strukturu a organizaci textu [59, 58]. Po provedení rozdělení byly dílčí části nahrazeny speciálními značkami (operace *tokenization*), jejíchž vyjmenování je uvedeno dole:

- `UNK` [`xxunk`] – značka zastupující slovo, které v aktuálním slovníku nevyskytuje;
- `PAD` [`xxpad`] – značka, která se přidává v případě přeskupení několika různých textu různé délky v jednom bloku;
- `BOS` [`xxbos`] – značka, která reprezentuje začátek textu v datové sadě;
- `FLD` [`xxfld`] – značka, která se používá pro oddělení dílčích částí, v případě, jestli je text sestaven z několika bloků;
- `TK_MAJ` [`xxmaj`] – token , který se přidává v případě, jestli slovo začíná velkým písmenem, například slovo „`Cat`“ bude označkováno jako „`xxmaj cat`“;
- `TK_UP` [`xxup`] – značka, která se přidává v případě, jestli slovo obsahuje všechny kapitálky, například slovní spojení „`I LOVE MY CAT`“ bude označkováno jako „`xxup i xxup love xxup my xxup cat`“;
- `TK_REP` [`xxrep`] – značka zastupující opakující se v textu symboly, syntaxe daného tokenu je následující: `xxrep` <*počet opakování*> <*symbol*>;
- `TK_WREP` [`xxwrep`] – značka zastupující opakující se v textu slova, syntaxe daného tokenu je následující: `xxwrep` <*počet opakování*> <*slovo*>.

Po provedení označkování datová sada vypadala následovně, viz obrázek číslo 3.17.

V dalším kroku došlo k provedení převodu označkovaného textu na celočíselné hodnoty, tj. k operaci *numericalization*. V rámci této etapy jsou slova z datové

idx	text
0	**xxbos xxup** re : **xxup** my **xxup** bengali ; eavesdropping program and the **xxmaj** patriot **xxmaj** act do n't understand the stakes in the war and into a **xxmaj** new **xxmaj** york **xxmaj** city high - rise on **xxmaj** october 11.the **xxmaj** national **xxmaj** transportation took the lives of five heroes , " **xxup** u.s. **xxmaj** forest **xxmaj** service **xxmaj** chaplain ...
1	**xxbos xxup** labradorite **xxup** formaldehyde ; after **xxmaj** sunday 's expected announcement of a verdict in the trial of former **xxmaj** iraqi leader statewide tour and plans to begin a 24-hour campaign blitz in **xxmaj** st. **xxmaj** louis on **xxmaj** friday told **xxup** cbs 's " **xxmaj** the **xxmaj** early **xxmaj** show " on **xxmaj** friday , before her brother **xxmaj** jason mckay 's funeral " **xxmaj** authorities were trying ...
2	**xxbos xxup** in **xxup** india , **xxup** people , **xxup** women **xxup** in **xxup** particular **xxup** either **xxup** find a **xxup** bus **xxup** or **xxup** an **xxup** auto **xxup** rickshaw **xxup** if **xxup** they **xxup** want **xxup** to **xxup** go **xxup** somewhere , **xxup** but **xxup** here **xxup** in **xxup** atlanta , **xxup** this **xxup** is **xxup** not **xxup** an **xxup** option . ; com will determine the distribution of the **xxmaj** general **xxmaj** contribution **xxmaj** fund ...

Obr. 3.17: Datová sada e-mailů po provedeném kroku *Tokenization*

sady postupně nahrazovány jejich pozicemi v používaném slovníku. Základní velikost slovní zásoby je z výkonnostního hlediska defaultně omezená na 60 000 slov [62]. V případě, jestli se převod slova nepodařil, bude slovo nahrazeno „neznámým“ tokenem, tj. `UNK` [59]. Problém s nahrazením nejčastěji může vzniknout v případě, jestli konkrétní slovo v aktuálním slovníku nevyskytuje. Dole, na obrázku číslo 3.18 je uveden výpis prvních dvaceti slov v použitém při provedení *Numericalization* slovníku a také výpis dvaceti přeložených slov první e-mailové entity v testovací datové sadě. Hodnoty v slovníku jsou seřazeny podle počtu výskytů v celé datové sadě.

```
# [Konzolový výstup]: dvacet prvních slov v aktuálně používaném slovníku
vocabluary(['xxunk', 'xxpad', 'xxbos', 'xxeos', 'xxfld', 'xxmaj', 'xxup', 'xxrep',
            'xxwrep', 'the', '.', ',', 'to', 'and', 'of', 'a', 'in', ':', '-', 'for', ...])

# [Konzolový výstup]: dvacet prvních hodnot prvního e-mailu po operaci Numericalization
array([    2,     6,    96,    17,     6,    86,     6, 21928,    52,  1060,
         314,    13,     9,     5,  1005,     5,   549,    59,    60,   513, ... ])
```

Obr. 3.18: Výpis spam slovníku a první zakódované elektronické zprávy

3.4.2 Vytvoření jazykového modelu

V první fáze procesu učení neuronové sítě byly provedeny všechny přípravné kroky, po jejichž splnění bylo možné přistoupit k trénování jazykového modelu neuronové sítě pomocí metody ULMFiT. V rámci podkapitoly číslo 3.4.1 byla předpřipravená datová sada, která posloužila základem pro inicializaci entity `TextLMDDataBunch`, obsahující vlastní slovník vytvořeny na základě elektronických zpráv a také zakódovaná do číselných hodnot e-mailová data. Tato entita v rámci této etapy poslouží stavebním prvkem na základě kterého bude specializován základní jazykový model postavený na slovníku `Wikitext 103`.

Je velmi důležité porozumět struktuře a principu fungování základního jazykového modelu, s cílem specializovat na konkrétní úlohu předtrénovanou neuronovou síť `AWD-LSTM`. Jak bylo zmíněno na začátku této podkapitoly je neuronová síť předtrénována na rozsáhlé datové sadě `Wikitext 103`[29]. Ve své podstatě se jedná o jazykový model, který sjednocuje kolekci více než 100 milionů tokenů, které byly vyextrahovány z veřejně dostupných článku na Wikipedia [41, 61]. Při odběru článků byl velký důraz kladen na jejich kvalitu, a proto jazykový model byl postaven pouze na takových článcích, které splňovaly podmínky, tzv. *Good*[30] a *Featured*[31]. Datová sada `Wikitext 103` je šířena pod licenci *Creative Commons Attribution-ShareAlike License*[32]. Oproti existujících datových sad, například `Penn Treebank` (`PTB`) a `Wikitext 2`, je vybrána sada mnohem rozsáhlejší [61]. V porovnání s datovou sadou `Penn Treebank` je `Wikitext 103` 110krát větší a oproti datové sadě `Wikitext 2` – 55krát větší [61]. V porovnání s `Penn Treebank` (`PTB`), sada `Wikitext 103` zachovává interpunkční znamínka, velikost písmen a vyskytující čísla. Při její trénování byly použity 28 000 článků a finální slovník obsahuje 267 735 entit [61].

V první fáze etapy vytvoření jazykového modelu přizpůsobeného na problematiku e-mailové komunikace pomocí metody `ULMFiT` došlo k načtení předtrénovaného modelu `AWD-LSTM` sítě. Z obrázku číslo 3.19 je patrné, že při načtení sítě `AWD-LSTM` byla specifikován základní jazykový model `Wikitext 103`, model zformované datové sady elektronických zpráv, architektura sítě a parametr LSTM multiple dropout (`drop_mult`). Tento parametr se používá pro regulaci velikosti dropout. V případě, jestli je model předtrénován (*over fitted*) je nutné tento parametr zvýšit, pokud dochází k opačné situaci, tj. *under fitting*, je nutné hodnotu `drop_mult` snížit [62].

[29]Datová sada jazykového modelu `Wikitext 103` je dostupná z URL: <`https://blog.einstein.ai/the-wikitext-long-term-dependency-language-modeling-dataset/`>.

[30]Informace o článcích typu *Good* na Wikipedia je dostpná z URL: <`https://en.wikipedia.org/wiki/Wikipedia:Good_articles`>.

[31]Informace o článcích typu *Featured* na Wikipedia je dostpná z URL: <`https://en.wikipedia.org/wiki/Wikipedia:Featured_articles`>.

[32]Znění licence je dostupné z URL: <`https://en.wikipedia.org/wiki/Wikipedia:Text_of_Creative_Commons_Attribution-ShareAlike_3.0_Unported_License`>.

Za zmínku stojí skutečnost, že při provedení trénování sítě byla použitá akcelerace výpočtů pomocí GPU. Samotný kód byl nejdřív testován ve virtuálním prostředí *Google Colaboratory*[33] a následně přemístěn na reálný hardware. Grafická karta použitá při vytvoření jazykového modelu a následně klasifikatoru byla `Nvidia Xp` s podporou architektury CUDA[34] a kapacitou paměti 12 196 MiB typu GDDR5X.

```
# Načtení předrenované na datové sadě Wikitext 103 AWD_LSTM sítě
awd_lstm_learner = language_model_learner(data=data_language_model,
arch=AWD_LSTM, pretrained_fnames=URLs.WT103, drop_mult=0.3)
```

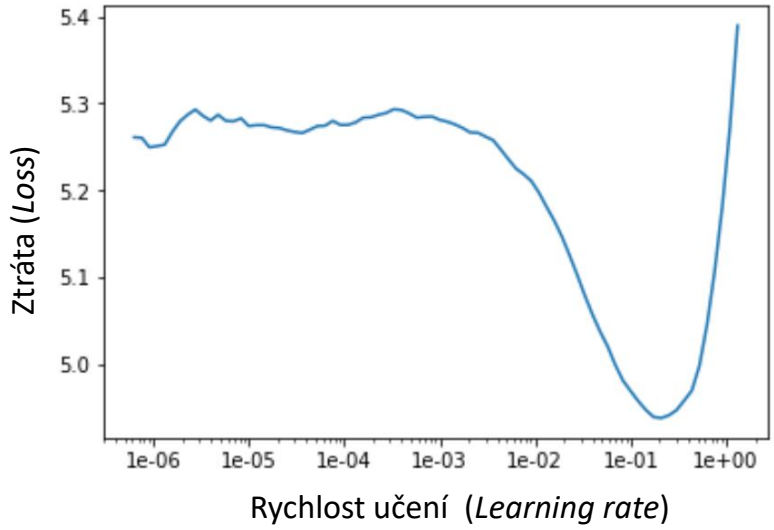
Obr. 3.19: Vytvoření instance a předtrénované neuronové sítě

Po provedeném načtení neuronové sítě a vytvoření instance `awd_lstm_learner` je nutné provést odhad rychlosti učení (*learning rate*) s cílem dosažení co nejrychlejší konvergenci k optimálnímu řešení [62]. Používaná knihovna `fast.ai` poskytuje speciální funkci, která provádí rychlý odhad možného rozpětí hodnoty *learning rate* pro konkrétní datovou sadu [59]. Princip fungování funkce spočívá v rozdělení původní datové sady na menší části a následné zvyšování rychlosti učení po zpracování každé z nich [41, 62]. Ve finále, po spouštění dané funkce došlo k vytvoření grafu (viz obrázek číslo 3.20), zohledňující vztah rychlosti učení a hodnoty účelové funkce, tj. `Loss function`. Z grafu je patrné, že v případě, jestli je *learning rate* roven jedné dochází k strmému nárůstu hodnoty `Loss` funkce – daný stav není možné považovat za žádoucí, a proto je odborníky doporučováno nastavovat takový *learning rate*, při kterém dochází ke klesání grafové křivky [59]. Pro budoucí učení jazykového modelu byla zvolena hodnota *learning rate* rovnající se $1 \cdot 10^{-2}$.

V dalším kroku bylo provedena jedná epocha trénování neuronové sítě. Výsledky je možné vidět v tabulce číslo 3.6. Hned na první pohled je vidět výhodu zvoleného *state-of-the-art* přístupu trénování předtrénované `AWD-LSTM` sítě, jelikož hned po první epoše úspěšnost jazykového modelu elektronických zpráv vytvořeného na bázi `AWD-LSTM` sítě převýšila 50 % a rovnala se 53,08 %. Pokud se zaměříme na technicky aspekt učení neuronové sítě, zjistíme, že během provedení první etapy trénování dochází k upřesnění vah pouze na posledních vrstvách neuronové sítě. Doba, nutná na provedení první epochy trénování jazykového modelu, zabrala 9 minut a 21 sekundu.

[33] Virtuální prostředí na vývoj aplikací je vyvinuto společností Google a je dostupné z URL: <`https://colab.research.google.com/`>.

[34] CUDA (*Compute Unified Device Architecture*) – architektura grafických karet vyvinuta společností `NVIDIA`, která poskytuje vlastní API a umožňuje provádět paralelizaci výpočtů.



Obr. 3.20: Závislost rychlosti učení a hodnoty `Loss` funkce u jazykového modelu

Trénovací ztrátovost (*Training loss*) se rovnala 2,9370 a validační ztrátovost (*Validation loss*) byla 2,7606. Z tabulky číslo 3.6 je také patrné, že hodnota *training loss* je vyšší něž *validation loss*, co fakticky znamená, že síť má prostor pro vylepšení a provedení dalších epoch trénování.

Tab. 3.6: Úspěšnost sítě `AWD-LSTM` po první epoše trénování

Epocha (*Epoch*)	Trénovací ztrátovost (*Training loss*)	Validační ztrátovost (*Validation loss*)	Přesnost (*Accuracy*)	Čas (*Time*)
0	2,9370	2,7606	**0,5308**	09 min 21 s

Pro dosažení lepších výsledků, je doporučeno udělat tzv. rozmrazení (*unfreezing*) celé neuronové sítě a provést několik dalších epoch učení se zachováním stejné rychlosti učení. Daný přístup umožní měnit váhy v rámci celé sítě, což může pozitivně ovlivnit konvergenci k řešení. V tabulce číslo 3.7 je vidět dosaženou přesnost jazykového modelu. Po provedených dvou epoch trénování, které celkově zabraly 21 minutu 45 sekund GPU času, finální přesnost (*accuracy*) jazykového modelu se rovnala 60,08 %. Přičemž hodnota *training loss* byla pokaždé vyšší než *valiation loss* a se rovnala: 2,8220 oproti 2,8034 v první etapě a 2,6115 oproti 2,3653 v druhé. V praxi pravděpodobnost naučeného jazykového modelu znamená, že natrénovaná neuronová síť dokáže s 60% pravděpodobností odhadnout následující slovo elektronické zprávy v případě provedení generování e-mailových zpráv.

Tab. 3.7: Úspěšnost sítě `AWD-LSTM` po provedení dalších epoch trénování

Epocha (*Epoch*)	Trénovací ztrátovost (*Training loss*)	Validační ztrátovost (*Validation loss*)	Přesnost (*Accuracy*)	Čas (*Time*)
0	2,8220	2,8034	**0,5372**	10 min 51 s
1	2,6115	2,3653	**0,6008**	10 min 54 s

Na konci dané etapy bylo provedeno testování vytvořeného jazykového modelu na datové sadě e-mailů. Výsledky testování jsou uvedeny na obrázku číslo 3.21. Z uvedeného je patrné, že při provedení testování modelu s výslednou pravděpodobností 60,08 % došlo k zadání počátečního textu (proměnná `TEXT`) vymyšlené zprávy. Cílem naučené sítě bylo provést doplnění zprávy a to takovým způsobem, aby byly splněny podmínky na maximální počet slov (proměnná `words`) a finální počet vygenerovaných e-mailů (proměnná `sentence`). Posouzení úspěšnosti doplňování zprávy necháme na pozorného čtenáře. Zajímavou vlastností jazykového modelu je schopnost dodržovat gramatická pravidla napsání vět a aktivně používat naučená z datové sady e-mailů slova. Pokud se pozorně zaměříme na vygenerovaný text je možné si všimnout, že se neuronová `AWD-LSTM` síť snaží při vytvoření zpráv zachovat stejnou strukturu dat použitých při její vytvoření. To znamená, že je předmět zprávy napsán kapitálkami a oddělen od samotného těla zprávy symbolem středníku.

3.4.3 Vytvoření klasifikátoru

Po splnění předchozí etapy a úspěšném natrénování jazykového modelu (viz podkapitola číslo 3.4.2), bylo možné přistoupit k vytvoření klasifikátoru spamových zpráv (viz obrázek číslo 3.15). Postup vytvoření klasifikátoru se bude skládat z několika kroků. V prvním kroku dojde k opakovanému načtení datové sady elektronických zpráv. Oproti načtení při modelování datové sady v první části (viz podkapitola číslo 3.4.1), v rámci daného kroku dojde k načtení přiřazených značek, které byly k elektronickým zprávám přidány v rámci procesu transformace dat (viz podkapitola číslo 3.2.5). Za připomínku je vhodné zmínit, že spamovým zprávám byla přiřazena hodnota `1` a důvěryhodným e-mailům - hodnota `0`. Po úspěšném načtení a modelování datové sady je možné přistoupit k vytvoření instance neuronové sítě sloužící pro provedení klasifikace. V rámci vytvoření této instance budou neuronové sítě jako argumenty předána instance označkované datové sady, a také kodér (*encoder*) z naučeného v rámci předchozí etapy jazykového modelu (viz podkapitola číslo 3.4.2). Po úspěšné realizaci výše zmíněných kroků proces učení sítě bude probíhat obdobným způsobem jako v případě trénování jazykového modelu.

```
0 ================================  [PRVNÍ ZPRÁVA]  ====================================
1 # Fáze testování vytvořeného jazykového modelu. Při testování dochází k zadání začátečného
2 # textu definovaného v proměnné @TEXT, maximálního počtu slov (@words) a počtu vygenerovaných
3 # zpráv (@sentence).
4 [In]:   TEXT = "RE: This message contains a"
5         words = 128
6         sentence = 1
7         print(awd_lstm_learner.predict(TEXT, words, temperature=0.5) for _ in range(sentence)))
8
9 # [Konzolový výstup]: Elektronická zpáva vygenerována sítí @awd_lstm_learner.
10 [Out]: "RE: This message contains a file named Attachment ID : NG - PRICE BOOK NUMBER FOR LESS
          THAN THE SPREAD OF THE NYMEX NOW ON THE DESK WE HAVE TO TRADE ON THE SIDE OF VIEW ; The
          following question is : Please approve the Deal Monthly Report to transactions by
          Monday, November 22 , 2001 . Please do not hesitate to call me at ext . 3 - 1411 .
          Thanks, John Arnold Original Message From : Arnold , John"
11  ============================================================================================
12
13  ================================  [DRUHÁ ZPRÁVA]  ====================================
14 [In]:   TEXT = "DISCOUNT:"
15         words = 128
16         sentence = 1
17         print(awd_lstm_learner.predict(TEXT, words, temperature=0.5) for _ in range(sentence)))
18
19 [Out]: "DISCOUNT: OUR ONLINE STORE COUPON ; We hope you enjoyed receiving this message . GIFT
          IDEAS  The Best Buy Now !  ? / AMAZON.COM Amazon.com Earth 's Biggest Selection Today !
          http : / .amazon.com - 0000 Please click here to unsubscribe from this list . Thanks for
          your assistance . Thanks , Eric xxbos RE : HE STRAIGHT ; and in the families , need ,
          said Jeanne Wade Evans , the San Bernardino later date . President Bush , working a
          southwest Missouri campaign crowd like a yell The"
20  ============================================================================================
```

Obr. 3.21: Testování naučené sítě `AWD-LSTM` na generování elektronických zpráv

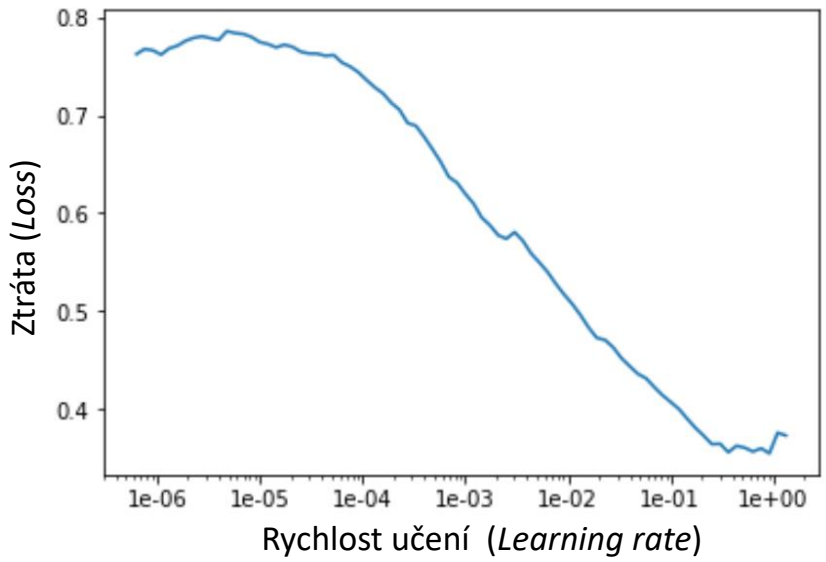
Proces modelování označkované datové sady a zároveň vytvoření instance budoucího klasifikátoru je názorně popsán na obrázku číslo 3.22. V rámci procesu modelování označkované datové sady lze vidět, že se používá vytvořený v průběhu první etapy slovník, předávaný proměnnou `data_language_model.train_ds.vocab`. Následně probíhá proces vytvoření instance klasifikátoru v jehož základu leží vymodelovaná datová sada elektronických zpráv (nízkoúrovňové detaily implementace této fáze jsou pro zjednodušení opuštěny, detailněji viz strojový kód) [63].

V případě vytvoření klasifikátoru, stejně jako při vytvoření jazykového modelu, pro odhad hodnoty rychlosti učení *learning rate* byla použita knihovna `fast.ai`. Po provedení rychlého odhadu závislosti hodnoty chybové funkce na *learning rate* (viz obrázek číslo 3.23), bylo stanoveno, že bude hodnota pro učení klasifikátoru e-mailových zpráv nastavena na $1 \cdot 10^{-3}$. Proces učení `ULMFiT` byl realizován pomocí metody `fit_one_cycle`, která je poskytována `fast.ai` a slouží k co nejrychlejšímu dosažení vysoké přesnosti neuronové sítě se zachováním kvality učení. Technicky je daná funkce realizována tak, že je při procesu učení hodnota *learning rate* dynamická. Za argument funkce `fit_one_cycle` se dosazuje tzv. maximální *learning rate*, jehož hodnota je dosažená v půlce času trénování jedné epochy.

```
# Vytvoření instance @data_classifier, která obsahuje označkovanou datovou sadu
# elektronických zpráv a také slovník použitý během tvorby jazykového modelu, který byl
# vytovřen v předchozích etapách.
data_classifier = TextClasDataBunch.from_df('./', train_df=train_frame,
valid_df=validation_frame, vocab=data_language_model.train_ds.vocab, bs=48)

# Proces uložení encoder vrstvy jazykového modelu @awd_lstm_learner
awd_lstm_learner.save_encoder('awd_lstm_learner_encoder')

# Vytvoření instance @awd_lstm_classifier, která přebírá datovou sadu elektronických zpráv
# a architekturu jazykového modelu
awd_lstm_classifier = text_classifier_learner(data_classifier, arch=AWD_LSTM,
drop_mult=0.5)

# Načtení encoder vrstvy natrénovaného jazykového modelu,
# která byla uložena v předchozím kroce
awd_lstm_classifier.load_encoder('awd_lstm_learner_encoder');
```

Obr. 3.22: Vytvoření instance klasifikátoru spamových zpráv

Na začátku je hodnota *learning rate* roste do svého maxima a následně po dosažení *maximum learning rate* začíná mírně klesat, a to až do skončení procesu trénování. Dynamická podoba rychlosti učení je motivována tím, že uprostřed svého učení parametr maximální rychlosti učení slouží regulačním mechanismem zabraňujícím přetrénovaní neuronové sítě [62, 59]. Následně při provedení sítí do procesu trénování vstupuje vnější proměnná *momentum*, která je spojená s rychlostí učení [66].



Obr. 3.23: Závislost rychlosti učení klasifikátoru a hodnoty `Loss` funkce

Při aplikování techniky `ULMFiT` je průběh funkce *momentum* přímo opačný k hodnotě *learning rate* [41]. Technicky to znamená, že při maximální výše hodnoty rychlosti učení je hodnota *momentum* minimální [62]. V praxi se ukázalo, že zavedení dané závislosti mezi hodnotou *learning rate* a *momentum* umožňuje trénovat neuronovou síť desetkrát rychleji, více o tomto vylepšení je možné dočíst z [62, 66].

V dalším kroku byla provedená první epocha trénování klasifikátoru. Výsledek první etapy učení je názorně zobrazen v tabulce číslo 3.8. Z ní je patrné, že akce-

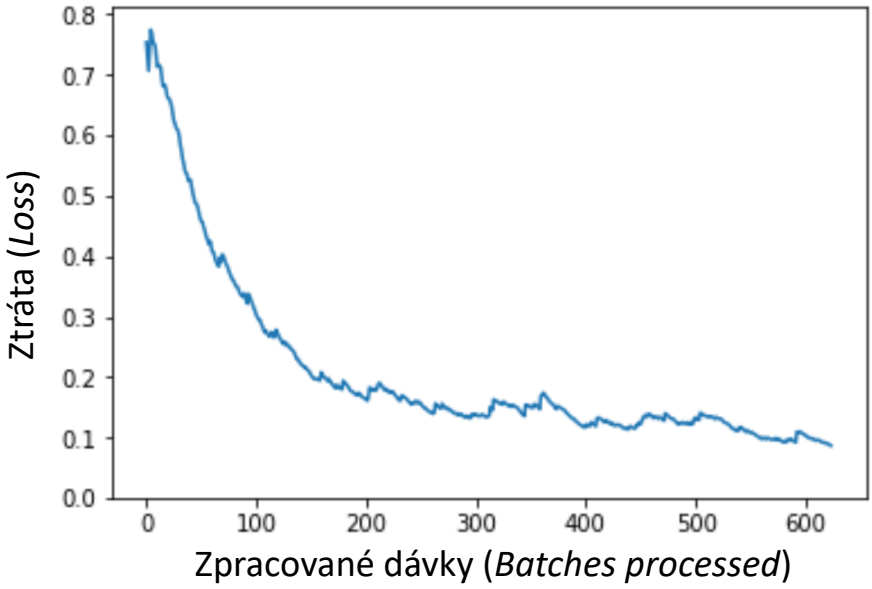
Tab. 3.8: Úspěšnost klasifikátoru po první epoše trénování

Epocha (*Epoch*)	Trénovací ztrátovost (*Training loss*)	Validační ztrátovost (*Validation loss*)	Přesnost (*Accuracy*)	Čas (*Time*)
0	0,1053	0,0328	**0,9888**	8 min 15 s

lerovaný pomocí GPU proces učení sítě, postavený na datové sadě e-mailů, celkově zabral 8 minut a 15 sekund s finální přesností 98,88 %. Zde je názorně vidět sílu použité techniky `ULMFiT` při provedení přeneseného učení na datové sadě e-mailů, která obsahovala pouze 30 000 vzorků pro trénování a 10 000 pro validaci. Tak vysoká přesnost modelu byla dosažena použitím předtrénované na rozsáhle datové sadě `Wikitext 103` neuronové sítě typu `AWD-LSTM`, která je v době napsání této práce nejvhodnějším a nejvíce slibujícím se základem pro vyřešení NLP problémů [41, 65]. Dalším důvodem argumentujícím vysokou přesnost modelu klasifikátoru, oproti, například. výsledkům při učení jazykového modelu je to, že v případě vytvoření jazykového modelu e-mailů se jednalo o proces učení sítě bez učitele (*unsupervised-learning*), který není doprovázen žádným mechanismem dohlížejícím nad procesem učení, a proto je síť nucena sama hledat závislosti v poskytovaných datech – proces *self-organizing* [62, 59, 65]. V případě řešení klasifikační úlohy byla nahrána označkovaná datová sada, která reprezentovala očekávané výsledky od učící se sítě. Z prezentované tabulky je také patrné, že hodnota *training loss* je vyšší než hodnota *validation loss* (0,1053 oproti 0,0328), což je slibné pro budoucí trénování.

Na obrázku číslo 3.24 je pomocí knihovny `fast.ai` vygenerován průběh účelové funkce během učení modelu klasifikátoru. Křivka má klesající charakter, co znamená, že se síť s každou další dávkou vstupných dat zlepšuje. Zároveň je z křivky patrné, že v dané etapě vývoje síť nedosahuje bodu nasycení (*saturation point*), při kterém jsou váhy vnitřních vrstev se blíží ke svým extrémům [28, 62].

V rámci diplomové práce bylo rozhodnuto pokračovat učení neuronové sítě `ULMFiT` a nezastavovat se na dosažené přesnosti v 98,58 %. Navíc, v případě metody `ULMFiT` v kombinaci s knihovou *fast.ai* je možné aplikovat pokročilejší metody učení sítě



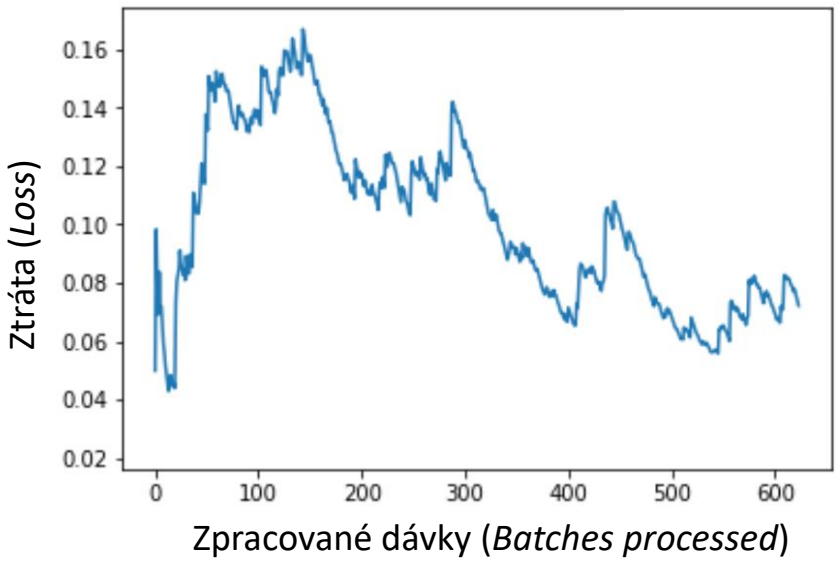
Obr. 3.24: Průběh účelové `Loss` funkce u klasifikátoru

spočívající například v použití metody `freeze_to` [41, 62]. Tato metoda umožňuje zamrazit pouze některé vnitřní vrstvy hluboké neuronové sítě, a tím docílit mnohem efektivnějšího přizpůsobení k vyřešení konkrétní úlohy [63]. V rámci diplomové práce bylo provedeno postupné rozmrazení posledních třech vrstev sítě pomocí metod `freeze_to(-2)` i `freeze_to(-3)` a aplikování na síť delších epoch přeneseného učení. Po provedení těchto kroků byla celá síť `AWD-LSTM` rozmrazena (`unfreeze`) a ještě víc dotrénována [62]. Výsledky výše zmíněných kroků pozitivně ovlivnily přesnost sítě, viz tabulka číslo 3.9. Z ni je patrné, že při dalším učení neuronové sítě, snižování rychlosti učení a aplikování pokročilejších mechanismů je možné dosáhnout lepších hodnot pravděpodobnosti.

Tab. 3.9: Finální přesnost klasifikátoru po přeneseném učení

Epocha (*Epoch*)	Trénovací ztrátovost (*Training loss*)	Validační ztrátovost (*Validation loss*)	Přesnost (*Accuracy*)	Čas (*Time*)
0	0,0529	0,0230	**0,9908**	7 min 32 s
1	0,0411	0,0180	**0,9938**	9 min 11 s
2	0,0367	0,0155	**0,9944**	9 min 22 s

První epocha[35] přeneseného učení byla provedena při rozmrazení posledních dvou vrstev. Doba trvání epochy byla 7 minut a 32 vteřiny, trénovací ztrátovost 0,0529 a validační – 0,0230. Přesnost se oproti předchozí etapě zvýšila o 0,2 % a rovnala se 99,08 %. Během druhé fáze trénování byly rozmrazeny poslední tři vrstvy. Doba trvání epochy byla 9 minut a 11 sekund. Trénovací ztrátovost byla 0,0411 a validační se rovnala 0,0180. Přesnos klasifikátoru se v dané fáze rovnala 99,38 %. Poslední epocha probíhala 9 minut a 22 sekundy při plném rozmrazení sítě a nastavenou minimální hodnotou *learning rate*. Absolutní přírůst byl oproti první epoše aplikování pokročilých technik roven 0,36 %. Po první inicializaci klasifikátoru se tedy finální přesnost zvýšila o $99,44 - 98,88 = 0,56$ %. Na obrázku 3.25 je znázorněna změna hodnoty `Loss` funkce během těchto etap.



Obr. 3.25: Průběh `Loss` funkce u klasifikátoru po dalším trénování

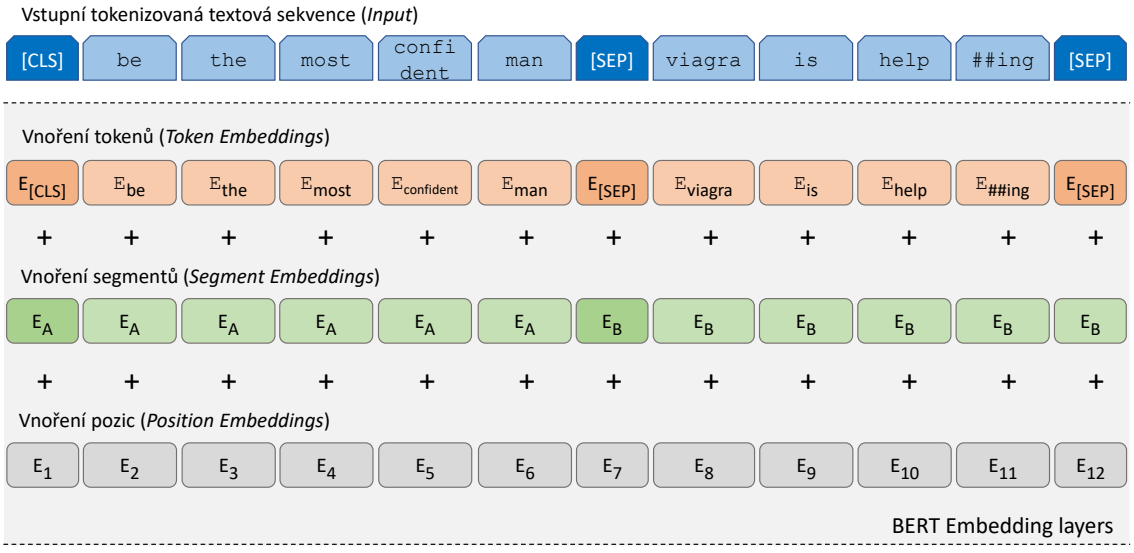
Po provedení trénování neuronové sítě `ULMFiT` sloužící pro klasifikaci e-mailových zpráv, je tato kapitola u svého logického konce. Vybraná neuronová síť byla pomocí techniky `ULMFiT` úspěšně naučena. Finální hodnota přesnosti klasifikátoru, který byl naučen na množině e-mailových zpráv, se rovnala 99,44 %. Podrobnější výsledky testování klasifikátoru viz kapitola číslo 4.

[35] Číslování epoch přeneseného učení bylo uděláno od nuly, proto pod první je myšlena `Epoch 0`.

3.5 Proces učení modelu Google BERT

Princip fungování a hlavní charakteristika modelu BERT byla provedena v podkapitole číslo 3.3.2. V této části diplomové práce budou detailněji popsány techniky vnoření slov používané v BERT, finální přizpůsobení dat k modelování neuronové sítě, hlavní princip a specifika přeneseného učení modelu a na závěr kapitoly bude určena přesnost a provedeno finální zhodnocení vytvořeného klasifikátoru.

Oproti síti ULMFiT pro své fungování používá BERT tři techniky vnoření, které lze vidět na obrázku 3.26. Jedná se o tři vrstvy: *Token Embeddings*, *Segment Embeddings* a *Position Embeddings* [43]. Všechny se nachází na vstupu neuronové sítě BERT, tj. před blokem transformer kodéru (viz obrázek číslo 3.26). Aplikování všech tří technik vnoření umožňuje síti dosahovat state-of-the-art výsledků a zachovat co nejvíce sémantických vazeb před vrstvami BERT kodérů. Z textu podkapitoly je patrné, že je na vstupní textovou sekvenci použitá funkce tokenizace, viz podkapitola 3.3.2. Maximální délka řetězce slov po tokenizaci je v síti BERT omezena na 512 tokenů [43]. Z obrázku vyplývá, že pro správné zpracování dat je nutné vstupní tokenizovanou sekvenci obohatit o speciální značky [CLS] a [SEP]. S cílem umožnění síti BERT řešit klasifikační úlohy je na začátek každé vstupní sekvence vnořena pomocná značka [CLS]. Navíc pro efektivnější trénování a možnost využití sítě na vyřešení úloh typu QA (*Question Answering*) a NSP (*Next Sentence Prediction*) umožňuje architektura BERT vytvořit tzv. větové vnoření (*sentence embedding*), a to přidáním značky [SEP] zastupující ukončení aktuální věty, kdy podle [43] pojem „věta“ nemusí být z lexikálního hlediska ekvivalentní s klasickou větou.



Obr. 3.26: Architektura vrstev vnoření v BERT [43]

Maximálně podporovaný počet `BERT` vět je omezen dvěma [43] a jejich identifikace uvnitř sítě je definována speciální binární maskou, viz 3.5.2. Pro každou větu platí, že vytvořené vnoření je stálé, ale liší se napříč dvěma větami [45].

První druh vnoření tokenů *Token Embeddings* je v případě modelu `BERT` podobný jiným moderním metodám neuronových sítí pro vyřešení NLP úloh. Jeho hlavním cílem je převedení konkrétního tokenu ze vstupní sekvence do dimenzionálně omezeného cílového prostoru. Konkrétně u modelu `BERT` se jedná o prostor s 768 dimenzemi, limitovaný slovníkem WordPiece obsahujícím 30 000 slov [45, 50].

Dalším druhem vnoření je *Segment Embeddings*, které primárně slouží k odhalení sémantické závislosti mezi dvěma vstupními větami již rozdělenými značkami `[CLS]` a `[SEP]` [50]. Realizace dané funkcionality je docílena přidáním k tokenizovaným vstupům binární masky, v níž je nulami pokrytá první část věty a jedničkami druhá, a to včetně poslední speciální značky `[SEP]`. V případě nutnosti doplnění textového řetězce do definované konstantní délky, jsou k masce přidány nulové *padding* bity, názorný příklad viz v podkapitole číslo 3.5.2.

Třetí a poslední druh vnoření *Position Embeddings* slouží k rozšíření funkcionality transformerů, které nerealizují kódování návaznosti vstupních tokenů. Vnoření pozic vstupních tokenů probíhá naučením sítě `BERT` vektorové reprezentaci každé pozice, čímž dochází k ovlivnění finální podoby vektoru v cílovém prostoru [43, 50].

3.5.1 Přizpůsobení datové sady k modelování

V prvním kroku stejně jako při vytvoření modelu `ULMFiT` bylo nutné provést přizpůsobení datové sady spamových zpráv před napojením na neuronovou síť. Na obrázku číslo 3.27 je vidět výpis kódu sloužícího k inicializaci objektu `BertDataProcessing`, která odpovídá za finální modelování dat a rozdělení datové sady na tři množiny, tj. trénovací, validační a testovací. V rámci daného kroku byly úpravy udělány v souladu s existujícími dokumenty, a to za účelem dosažení nejlepších výsledků, [43, 48].

```
1  # Vytvoření objektu obsahujícího předzpracované datové sady e-mailů.
2  # Hodnota @seed je vložena s cílem znáhodnění dat při rozdělení. Funkce
3  # @BertDataProcessing zpracovává e-maily takovým způsobem, aby její výstup
4  # bylo možné bezproblemově napojit na model BERT.
5  bertDataProcessing = self.BertDataProcessing(path_spam, path_ham, seed=99)
6
7  # Naplnění datových rámců pro budoucí použití.
8  bert_train = bertDataProcessing.get_trainframe()
9  bert_validation = bertDataProcessing.get_validationframe()
10 bert_test = bertDataProcessing.get_testframe()
```

Obr. 3.27: Finální přizpůsobení datové sady k modelování

Navíc při zpracování dat byly v potaz vzaty doporučení zveřejněné v oficiálním BERT Python Notebook[36]. Data po zpracování byla rozdělena do dvou sloupců, viz 3.27. První sloupec labels obsahoval značky přiražené k příslušným e-mailovým zprávám, kde značka 1 byla přiřazena ke spamovým zprávám a značka 0 k důvěryhodným. Druhý sloupec text byl zastoupen samotnými zřetězenými texty e-mailů. Oproti úpravě pro síť ULMFiT byl text zpráv převeden na malá písmena. Na začátek e-mailového řetězce byl přiřazen klasifikační tag [CLS], viz 3.3.2. Navíc na základě testování bylo rozhodnuto, že síť ukazuje lepší výsledky pokud předmět zprávy a jeho textová část jsou pomocí značky [SEP] rozděleny do zvláštních vět. Na obrázku číslo 3.28 je možné vidět výpis trénovacího datového rámce po modelování.

```
================================ TRAIN DATAFRAME ================================
       labels                                                  text
0           1  [CLS] just few weeks away from a lean, toned,  ...
1           1  [CLS] be the most confident man in town [SEP]  ...
2           1  [CLS] start off the new year in style! [SEP]   ...
3           1  [CLS] i do not bid thee beg my life, good lad, ...
4           1  [CLS] take care [SEP] dear sir, we are happy t ...
...       ...                                                 ...
29995       0  [CLS] re: re-assignment of transmission from t ...
29996       0  [CLS] you are in big trouble [SEP] you are in  ...
29997       1  [CLS] see drone x pro in action [SEP] best aff ...
29998       0  [CLS] it resources [SEP] this makes great sens ...
29999       0  [CLS] transmission [SEP] tjae, please find a r ...

[30000 rows x 2 columns]
============================== END TRAIN DATAFRAME ==============================
```

Obr. 3.28: Podoba trénovacího datového BERT rámce po přizpůsobení

3.5.2 Aplikování přeneseného učení na vytvoření klasifikátoru

V další etapě, tj. po vytvoření datových rámců s e-maily bylo přistoupeno k samotnému učení neuronové sítě BERT . Zjednodušeně je možné daný proces rozdělit do dvou logicky navazujících etap. V první etapě dochází k načtení veřejně dostupného BERT tokenizátoru, který datovou sadu připravenou v rámci předchozí kapitoly převede do vyhovující pro síť podoby. Konkrétně se jedná o převod vstupních vět na tokeny, vytvoření příslušných *segment* a *attention* masek, numerikalizaci (*numericalization*) vstupů a aplikování tři technik vnoření, tj. *Token Embeddings*, *Segment Embeddings* a *Position Embeddings*, s cílem co nejpřesnějšího odražení tokenů do

[36]Vzorový příklad implementace BERT a krátké komentáře k hlavním principům jeho fungování jsou dostupné z oficiálního URL: <https://colab.research.google.com/drive/1ywsvwO6thOVOrfagjjfuxEf6xVRxbUNO>.

vícedimenzionálního prostoru. Ve druhé etapě by došlo k napojení vstupních vektorových tenzorů na neuronovou síť a aplikování přeneseného učení se zaměřením na vyřešení úlohy binární klasifikace [43].

Pro úspěšnou realizaci první etapy síť `BERT` vyžaduje poskytnutí následujících formátovaných datových vstupů, detailněji viz podkapitolu 3.3.2:

- `input_id` – sekvence celých čísel reprezentující pozice vstupních tokenů v používaném `BERT` slovníku;
- `labels` – přiražené značky zastupující cílové kategorie. V případě této úlohy se jedná o binární klasifikaci, proto mohou byt jednotlivé prvky sekvence zastoupeny pouze jedničkami a nulami;
- `segment_mask` – nepovinný parametr, pomocí kterého je možné identifikovat, zda se v sekvenci `input_id` jedná o první anebo druhou `BERT` větu. První věta a padding tokeny jsou zastoupeny nulovými prvky, druhá věta jedničkami;
- `attention_mask` – nepovinný parametr, sekvence nul a jedniček umožňující odlišit, zda se jedná o originální token, anebo uměle doplněný, tj. *padding*.

Na obrázku číslo 3.29 je znázorněn zjednodušený výpis vytvořeného kódu.

```
# Inicializace a nahrání tokenizatoru @tokenizer z modelu bert-base-uncased.
# Přepnutí @tokenizer do režimu zpracování textu v malém písmu.
tokenizer = BertTokenizer.from_pretrained('bert-base-uncased', do_lower_case=True)

# Naplnění proměných @tokenized_train, @tokenized_valid, @tokenized_test tokeny,
# které byly získány z původní datové sady, konkrétně ze sloupce @text.values.
# Pro zjednodušení a lepší pochopení principů fungování daného procesu byl kód
# oproti originálu zjednodušen.
tokenized_train = [tokenizer.tokenize(w) for w in bert_train.text.values]
tokenized_valid = [tokenizer.tokenize(w) for w in bert_valid.text.values]
tokenized_test = [tokenizer.tokenize(w) for w in bert_test.text.values]

# Zpracování trénovacího, testovacího a validačního datového rámce a vytvoření
# segmentní masky s cílem provedení Segment Embeddings. V tomto kroku dochází k
# omezení sekvence tokenů na maximální délku (@MAX_LEN) a paddingu. Pro zjednodušení
# a lepší pochopení principů fungování daného procesu byl kód zjednodušen.
segments_train = self.process_tokenized_text(MAX_LEN, tokenized_train, segments_train)
segments_test = self.process_tokenized_text(MAX_LEN, tokenized_test, segments_test)
segments_valid = self.process_tokenized_text(MAX_LEN, tokenized_valid, segments_valid)

# Provedení numerikalizace datových rámců @tokenized_train, @tokenized_test,
# @tokenized_valid. Vytvořené rámce budou použité během fine-tunning BERT.
input_ids_train = [tokenizer.convert_tokens_to_ids(t) for t in tokenized_train];
input_ids_test = [tokenizer.convert_tokens_to_ids(t) for t in tokenized_test];
input_ids_valid = [tokenizer.convert_tokens_to_ids(t) for t in tokenized_valid];

# Vytvoření tzv. attention masky s cílem umožnění BERT odlišit původní a umělé
# doplněnou sekvenci (v rámci provedení operace padding do délky @MAX_LEN.
attention_train = self.create_attention_masks(input_ids_train, attention_train)
attention_test = self.create_attention_masks(input_ids_test, attention_test)
attention_valid = self.create_attention_masks(input_ids_valid, attention_valid)
```

Obr. 3.29: Princip tokenizace, numerikalizace a vytvoření masek pro datové rámce

Po jeho spuštění dochází ke stažení a inicializaci modelu tokenizátoru pro neuronovou síť `bert-base-uncased`. Inicializovaný model `BERT` obsahuje 12 vrstev, 768 skrytých prvků, 12 *self-attention* hlaviček a 110 milionů optimalizačních parametrů, detailněji viz 3.3.2. Výběr padl na tento model neuronové sítě z důvodu nejmenší paměťové složitosti pro provedení operace inicializace a trénování. Existující omezení byla daná paměťovou kapacitou používané grafické karty. Stejně jako při zprovoznění modelu `ULMFiT` byla síť `BERT` nejdříve otestována ve virtuálním prostředí *Google Colaboratory*[37] a následně, po úspěšném dokončení dané fáze, byla přenesena na reálný hardware. Konkrétně se jednalo o grafickou kartu `Nvidia Xp` s podporou architektury CUDA a kapacitou paměti 12 196 MiB typu GDDR5X. Jak je možné zjistit z oficiální dokumentace spuštění modelu `bert-large-uncased`[38] by vyžadovalo minimálně dvě mezi sebou propojené grafické karty podobné úrovně [48]. V případě modelu `bert-base-uncased` používaného v dané diplomové práci, inicializace a učení modelu bylo možné pouze po provedení řady optimalizací a nastavení minimálních parametrů pro učení, viz dále.

Z obrázku 3.29 je patrné, že v prvé řadě byly vstupní datové rámce tokenizovány, a následně použity na trénování, testování a validaci. Konzolový výstup po provedené tokenizaci lze vidět na dalším obrázku (viz 3.30). Následně pomocí naprogramované funkce `process_tokenized_text` probíhá pro jednotlivé tokenizované sekvence vytvoření *segment* masek. Jak bylo uvedeno výše architektura `BERT` vyžaduje provedení zarovnání všech sekvencí do bloků s pevně danou velikostí, která je omezená parametrem `MAX_LEN`. Funkce `process_tokenized_text` provádí analýzu délek textových sekvencí a jejich následující úpravu. V případě, kdy je délka analyzované sekvence vyšší než hodnota proměnné `MAX_LEN`, bude délka řetězce oříznuta na maximálně povolenou hodnotu, tj. `MAX_LEN - 1`[39]. Pokud délka vstupu bude menší než hodnota `MAX_LEN`, bude vstupní sekvence doplněná *padding* bity a daná změna se promítne do příslušné masky pozornosti (*attention mask*). V dalším kroku jsou tokenizované texotvé hodnoty převedeny do celočíselného formátu pomocí vnitřní funkce tokenizéru `convert_tokens_to_ids`, tzn. jsou podvržené operaci numerikalizace. Příslušné číselné hodnoty jsou přiraženy na základě vnitřního `BERT` slovníku a odpovídají pozicím konkrétních slov, viz kapitola 3.3.2. V posledním kroku jsou do proměnných `attention_train`, `attention_test` a `attention_valid` načteny spočítané masky pozornosti, viz 3.29.

[37] Virtuální prostředí na vývoj aplikací je vyvinuto společností Google a je dostupné z URL: <`https://colab.research.google.com/`>.

[38] Samozřejmé není vyloučeno, že by použití sítě `bert-large-uncased` umožnilo dosáhnout lepších finálních výsledků přesnosti, jelikož má větší množství vrstev kodérů a optimalizačních parametrů, které by dovolily odhalit více skrytých závislostí při vyřešení této NLP úlohy.

[39] Odečíst jedničku je nutné proto, že se na konci každé tokenizované sekvence musí nacházet speciální značka `[SEP]` oznamující konec `BERT` věty.

Na obrázku číslo 3.30 lze názorně vidět konzolové výpisy všech zmíněných výše datových struktur. První konzolový výstup reprezentuje datový rámec e-mailů použitý na trénování sítě `BERT` , a to hned po provedení jeho tokenizace. Tři další výpisy odráží náplň proměnných, které budou přímo napojeny na inicializovanou neuronovou síť. V daném okamžiku je důležité zmínit, že všechny tři datové struktury byly normalizovány a jejich délky byly omezeny na `MAX_LEN`. Dané zpracování z nich umožnilo vytvořit tenzory, které jsou si velikostně rovny. Pro trénovací datový rámec byla velikost tří příslušných tenzorů $30\,000 \times 512$, kde je 30 000 číslo entit v trénovací datové sadě a hodnota 512 odpovídá hodnotě `MAX_LEN`. První tenzor zastupuje převedené textové tokeny na jejich číselná vyjádření. Pro srovnání byly na zobrazení vybrány první dvě sekvence, jejichž délky nepřesahovaly hodnotu `MAX_LEN` a jedna sekvence (na obrázku je zastoupena posledním prvkem), která byla uříznuta. Druhý tenzor zastupuje masky pro odlišení první a druhé věty, případně použitého *padding* (poslední nulové bity v prvních dvou sekvencích). Poslední tenzor zastupuje vytvořené *padding* masky pro odlišení originálního obsahu od doplněného.

```
# [Konzolový výstup]: Tokenizovaná vstupní sekvence spamových zpráv.
  [[ '[CLS]', 'just', 'few',  ... ,  'reserved', '.', '[SEP]'],
   [ '[CLS]', 'be', 'the', 'most',  ... ,    'lower', '[SEP]'],
     ... ,
   [ '[CLS]', 'transmission', '[SEP]', ... , '##ng',  '[SEP]']]

# ================================================================

# [Konzolovvý výstup]: Vstupní sekvence po numerikalizaci a paddingu.
  tensor([[  101,  2074,  2261,  ... ,     0,     0,     0],
          [  101,  2022,  1996,  ... ,     0,     0,     0],
          ... ,
          [  101,  6726,   102,  ... ,   432,   532,   102]])

# [Konzolovvý výstup]: Segmentní maska pro Segment Embeddings.
  tensor([[0, 0, 0,  ... , 0, 0, 0],
          [0, 0, 0,  ... , 0, 0, 0],
          ... ,
          [0, 0, 0,  ... , 1, 1, 1]])

# [Konzolovvý výstup]: Padding mask vstupní sekvence @MAX_LEN:
  tensor([[1, 1, 1,  ... , 0, 0, 0],
          [1, 1, 1,  ... , 0, 0, 0],
          ... ,
          [1, 1, 1,  ... , 1, 1, 1]])
```

Obr. 3.30: Výpis trénovacích tenzorů před napojením na neuronovou síť `BERT`

Po provedení zpracování bylo přistoupeno k přenesenému učení `BERT` sítě. Samotný proces trénování se principiálně velmi blíží k trénování ULMFiT. Stejně jako v předchozím případě se jedná o časově relativně nenáročnou úlohu oproti klasickému přístupu, kde dochází k trénování neuronové sítě od základů. V případě `BERT`

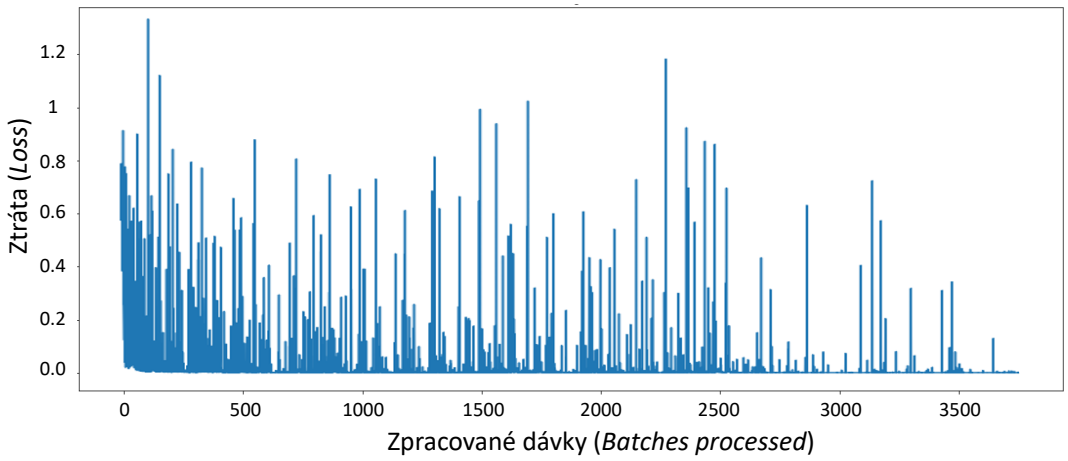
stačí na výstupu neuronové sítě přidat vrstvu specifickou pro konkrétní NLP úlohu a provést její trénování. Laicky řečeno kvůli existující generalizaci modelu je 95 % existujících vrstev už předtrénováno. Pro vyřešení problému klasifikace nevyžádané pošty bude tedy nutné poskytnout modelu co nejkvalitnější datovou sadu a aplikováním technik přeneseného učení dotrénovat zbylých 5 % sítě.

Po provedení různých pokusů učení na stejných datech, bylo nejlepších výsledků dosaženo po nastavení následujících konfiguračních parametrů:

- `MAX_LEN = 512` – maximální délka vstupní sekvence;
- `LEARNING_RATE = 2e-5` – rychlost učení sítě;
- `BATCH_SIZE = 8` – množství entit najednou předaných do sítě na zpracování;
- `NUMBER_OF_EPOCHS = 2` – počet epoch přeneseného učení `BERT`;
- `DROPOUT = 0.1` – hodnota droupout během učení.

Pozorný čtenář si může všimnout výraznou změnu oproti modelu ULMFiT. Konkrétně se jedná o snížení velikosti parametru `BATCH_SIZE`, který byl snížen z 48 na 8. Daný krok byl učiněn z důvodu vyšších paměťových nároků sítě `BERT`. Proto je v případě provedení přímého srovnání důležité brát v potaz, že se množství dávek, neboli *batches*, bude při stejném množství trénovacích dat lišit.

Na obrázku číslo 3.31 je názorně vidět pokles hodnoty účelové fitness funkce při postupném učení sítě. Z obrázku je také patrné, že trénovací datová sada obsahující 30 000 e-mailových entit při `BATCH_SIZE = 8` byla rozdělena do více než 7000 dávek.



Obr. 3.31: Průběh `Loss` funkce `BERT` klasifikátoru

Po aplikování technik přeneseného učení byl model úspěšně dotrénován, viz tabulka 3.10. První epocha učení `BERT` zabrala 38 minut a 44 sekundy. Během ní se hodnota `Loss` funkce rovnala 0, 0330, přičemž přesnost modelu byla 99, 52 %, což je zřetelně vyšší ukazatel, než u srovnávaného modelu `ULMFiT` po prvním cyklu

trénování. Do viditelných nevýhod hned po první epoše trénování je možné zařadit srovnatelně větší dobu nutnou na provedení jednotlivé etapy. Dané zpoždění může být odůvodněno jak vetší asymptotickou složitostí `BERT`, tak i menší hodnotou `BATCH_SIZE`. Druhá a poslední epocha zabrala 38 minut a 46 sekund, trénovací ztrátovost byla $0,0066$ a finální přesnost se na validačních datech rovnala $99,66\,\%$.

Tab. 3.10: Vyhodnocení procesu učení modelu sítě `BERT`

Epocha (*Epoch*)	Ztrátovost (*loss*)	Validační přesnost (*Validation accuracy*)	Čas (*Time*)
0	0,0330	**0,9952**	38 min 44 s
1	0,0066	**0,9966**	38 min 46 s

Z obdržených výsledků je vidět, že aplikování transformerů a *attention-based* přístupu umožnilo za dvě epochy trénování dosáhnout *state-of-the-art* výsledků. Podrobné otestování naučeného modelu na nachystaných testovacích datech bude provedeno v podkapitole číslo 4.1. Po jeho provedení bude možné jednoznačně porovnat modely natrénované v rámci diplomové práce a udělat odpovídající závěry.

3.6 Proces učení modelu `Google Brain XLNet`

V podkapitole číslo 3.3.3 byl popsán princip fungování hluboké sítě `XLNet`, její mechanismy pro analýzu textových sekvencí a vylepšení oproti sítím `BERT` a `ULMFiT`. V této podkapitole bude popsáno finální přizpůsobení dat pro modelování, proces učení sítě a dosažené výsledky při aplikování na úlohu filtrování nevyžádané pošty.

Při vývoji se autoři `XLNet` snažili vycházet z existujícího modelu `BERT` a umožnit programátorům co nejjednodušší integraci datové sady do neuronové sítě a následné provedení samotného trénování. Samozřejmě není možné konstatovat, že princip zpracování dat a jejich napojení na síť jsou stejné jako v případě `BERT`. Odlišnosti modelů `BERT` a `XLNet` spočívají v lišícím se principu vytvoření zastupujících sekvencí a v různých technikách vytvoření *word-embedding*. Model sítě `XLNet` jako svůj slovník používá SentencePiece obsahující 32 000 slov. Slovní zásoba `BERT` je zastoupena slovníkem WordPiece obsahující 30 000 slov. Podobnost modelů se kotví v použití speciálních značek a také v převedení vstupních tokenů na čtyři celočíselné sekvence naplňující proměnné `input_id`, `labels`, `segment_mask` a `attention_mask`, viz dále.

Pro vyřešení úlohy definované v zadání této práce byla zvolena menší verze sítě `XLNet`, a to z důvodu nedostačujících paměťových kapacit grafické karty pro učení větší verzi sítě, tj. `XLNet-Large`. Obdobně jako při zprovoznění modelu `BERT` byla síť

`XLNet` nejdříve otestována ve virtuálním prostředí *Google Colaboratory*[40] a následně, po úspěšném dokončení dané fáze, byla vybraná neuronová síť přenesena na reálný hardware, který byl zastoupen grafickou kartou `Nvidia Xp` s podporou architektury CUDA a kapacitou paměti 12 196 MiB typu GDDR5X.

3.6.1 Přizpůsobení datové sady k modelování

Po provedené transformaci dat, viz podkapitola číslo 3.2.5, bylo nutné načíst a zpracovat data takovým způsobem, aby v budoucnu bylo možné co nejjednodušeji provést jejich tokenizaci, numerikalizaci a vytvořit vnoření slov (*word embedding*). Jelikož pro trénování byl zvolen model `XLNet-Base`, který je citlivý na velikost písma, byla existující třída na modelování dat přizpůsobena takovým způsobem, aby poskytovala na výstupu textovou sekvenci ohraničenou `XLNet` značkami a navíc prováděla požadované formátování textu. Vytvořená třída `XLNetDataProcessing` (viz zdrojový kód práce) převáděla předmět zprávy elektronických zpráv na velká písmena, zpracovávala URL odkazy a přidávala speciální značky, které `XLNet` používá pro rozpoznání vstupních vět. V daném kroku je důležité zmínit, že pro svoji činnost `XLNet` vyžaduje následující druhy značek:

- `<unk>` – neznámý token, jehož převod není možné pomocí slovníku zajistit;
- `<cls>` – značka sloužící k označení začátku textové sekvence při klasifikaci;
- `<sep>` – značka, která je přidávána na konci každé věty k jejich rozdělení;
- `<eod>` – speciální značka `XLNet` na konci sekvence používána tokenizátorem.

Na obrázku číslo 3.32 je prezentováno vytvoření objektu sloužícího pro finální zpracování a generování trénovacího, testovacího a validačního datového rámce.

```
# Vytvoření objektu obsahujícího předzpracované datové sady e-mailů.
# Hodnota @seed je vložena s cílem znáhodnění dat při rozdělení. Funkce
# @XLNetDataProcessing zpracovává e-maily takovým způsobem, aby její výstup
# bylo možné bezproblemově napojit na model sítě XLNet.
xlnetDataProcessing = self.XLNetDataProcessing(path_spam, path_ham, seed=99)

# Naplnění datových rámců pro budoucí použití v modelu XLNet.
xlnet_train = xlnetDataProcessing.get_trainframe()
xlnet_validation = xlnetDataProcessing.get_validationframe()
xlnet_test = xlnetDataProcessing.get_testframe()
```

Obr. 3.32: Finální příprava datové sady k modelování

[40] Virtuální prostředí pro vývoj aplikací je vyvinuto společností Google a je dostupné z URL: <`https://colab.research.google.com/`>.

Kód byl navržen s motivací co nejlepšího pochopení pozorným čtenářem, a to takovým způsobem, aby se syntakticky co nejvíce přibližoval kódu použitému při finálním přizpůsobení elektronických zpráv v BERT a ULMFiT.

Hluboká neuronová síť XLNet stejně jako BERT podporuje rozdělení textu do dvou vět (podobně viz podkapitola 2.4). Rozdělení je docíleno aplikováním značky <sep> a změnou segment_mask během dalších fází zpracování textu [51]. Na rozdíl od BERT, síť XLNet nerealizuje mechanismus NLP, neboli *Next Sentence Prediction*. Rozdělení na věty tedy slouží pro efektivnější trénování sítě a přesnější odhalení vazeb ve vstupním textu, detailněji viz [51]. Během testování různých parametrů a nastavení bylo zjištěno, že hluboká síť XLNet dosahuje lepších výsledků[41], když předmět e-mailu a jeho text jsou rozděleny do různých XLNet vět.

3.6.2 Aplikování přeneseného učení na vytvoření klasifikátoru

Po dokončení předchozí etapy bylo přistoupeno k inicializaci modelu neuronové sítě a aplikaci technik přeneseného učení na vytvoření klasifikátoru spamových zpráv. Etapově se daný proces velmi přibližoval ke stejnojmennému procesu u sítě BERT. V první řadě byla vytvořena Python třída process_tokenized_text sloužící pro převedení textu na tokeny. Převod byl realizován tokenizátorem, již poskytovaným knihovnou PyTorch. Následně třída poskytovala rozhraní pro naplnění proměnných input_id, labels, segment_mask a attention_mask, a to na základě tokenizované textové sekvence. Ačkoliv se výstupy pro síť XLNet skládaly ze stejného množství sekvencí jako v případě sítě BERT (viz 3.5.2), logika jejich naplnění se lišila. Níže je uveden seznam proměnných vytvořených ze vstupní sekvence a jsou popsány rozdíly v postupech jejich naplnění:

- segment_mask – sekvence celých čísel odrážející logické rozdělení vstupní sekvence na věty a speciální značky; Oproti síti BERT se nejedná o jednoduchou sekvenci nul a jedniček, ale pěti čísel v rozsahu nula až čtyři. Jejich použití se liší v závislosti na nastavených parametrech sítě XLNet a množství používaných vět. V případě vytvoření pokročilého mechanismu na filtrování nevyžádaných zpráv, došlo ke vnoření všech značek do masky segmentů. Nulové prvky zastupovaly označení první XLNet věty, tj. SEG_ID_A, do níž byl zahrnut předmět e-mailové zprávy. Jednotkové prvky pokrývaly druhou XLNet větu, která zastupovala textové tělo e-mailů a označovala se jako SEG_ID_B. Na místech použití speciálního tokenu <cls>, tj. na začátku tokenizované sekvence, byla do masky segmentů vložená značka SEG_ID_CLS = 2. Trojkové prvky zastupovaly hraniční značky <sep>. *Padding* prvky byly v masce segmentů označeny SEG_ID_PAD = 4. Podobu vytvořené masky lze vidět na obrázku číslo 3.33.

[41] Absolutní přírůst přesnosti pro tyto dva způsoby byl u modelu XLNet 1-1,5 %.

- `attention_mask` – sekvence nul a jedniček umožňující odlišit, zda se jedná o reálná data vstupního tokenu, anebo data uměle doplněná do `MAX_LEN` pomocí *padding*. Oproti `BERT` je princip generování této masky odlišný. Doplnění *padding* bitů se provádí na začátek sekvence, nikoliv na její konec. Navíc jedničkové bity zastupují doplněné tokeny a nulové pokrývají množinu reálných tokenizovaných vstupů. Podobu masky pozornosti lze vidět na obrázku číslo 3.33.
- `input_id` – sekvence celých čísel reprezentující pozice tokenů v používané slovní zásobě `XLNet`. Princip naplnění je stejný jako v případě `BERT`, tzn. pomocí vestavěné funkce `XLNet` tokenizátoru `encode`.
- `labels` – celočíselné hodnoty, které zastupují kategorie přiřazené e-mailovým zprávám. Logika přiřazení je stejná jak i v případě `ULMFiT` a `BERT`;

Na obrázku číslo 3.33 je uveden výpis výše uvedených proměnných, které byly vytvořeny a naplněny pomocí `Python` třídy `process_tokenized_text`. Délka všech proměnných byla omezená na délku maximálně podporovanou sítí `XLNet`, tj. na 512. Množství entit pro trénovací množinu bylo zachováno a jako v případě `BERT` a `ULMFiT` se rovnalo 30 000. Velikostně se tedy tensory rovnaly $30\,000 \times 512$.
Mezi charakterní vlastnosti tensorů je možné zahrnout doplnění *padding* bitů na začátek sekvencí. Inverzní masky jsou oproti `BERT` také inverzní, viz obrázek 3.33. Segmentní masky obsahují hodnoty od 0 do 4, viz popis výše.

Po převodu dat na numerické tensory bylo možné přistoupit k inicializaci sítě a aplikování technik přeneseného učení. V daném okamžiku je dobré zmínit, že oproti jiným technikám určení nejlepších parametrů pro trénování sítě zabralo největší množství času. Při aplikování doporučených parametrů na problém klasifikace e-mailů nebyly výsledky sítě vynikající, v některých případech docházelo k přeučení sítě a v jiných k nízké finální přesnosti na testovacích datech. Modifikované parametry lze tedy rozdělit do dvou skupin.

První skupina obsahovala ty, jejichž hodnoty během různých pokusů přeneseného učení zůstávaly neměnnými. Do této skupiny patřily níže uvedené proměnné:

- `MAX_LEN = 512` – maximální délka vstupní sekvence v `XLNet` byla nastavená na maximální s účelem dosažení co nejvyšší přesnosti během zpracování zpráv [52];
- `BATCH_SIZE = 8` – množství entit najednou předaných do sítě na zpracování. Proměnná byla nastavena podle doporučení dostupných z [52].

Druhá skupina zastupovala proměnné, jejichž hodnoty byly během optimalizace parametrů upravovány. Do této skupiny patřily uvedené níže proměnné[42]:

- `LEARNING_RATE = 2e-5` – rychlost učení sítě (*doporučená hodnota je 3e-5*) [55];
- `NUMBER_OF_EPOCHS = 2` – počet epoch přeneseného učení `XLNet`;

[42]Za zmínění stojí i skutečnost, že hodnoty uvedené u zmíněných v seznamu proměnných byly použity při nejvíce úspěšném učení neuronové sítě.

```
# [Konzolovvý výstup]: Vstupní sekvence po numerikalizaci a paddingu:
  tensor([[    0,     0,     0,  ...,     3,     4,     3],
          [    0,     0,     0,  ...,     3,     4,     3],
          [    0,     0,     0,  ...,     3,     4,     3],
          ...,
          [   17, 23999,    17,  ..., 10643,     4,     3],
          [    0,     0,     0,  ...,     3,     4,     3],
          [    0,     0,     0,  ...,     3,     4,     3]])

# [Konzolovvý výstup]: Kategorizační značky - spam (1) anebo ham (0):
  tensor([1, 1, 1,  ..., 1, 0, 0])

# [Konzolovvý výstup]: Attention maska velikosti @MAX_LEN:
  tensor([[1, 1, 1,  ..., 0, 0, 0],
          [1, 1, 1,  ..., 0, 0, 0],
          [1, 1, 1,  ..., 0, 0, 0],
          ...,
          [0, 0, 0,  ..., 0, 0, 0],
          [1, 1, 1,  ..., 0, 0, 0],
          [1, 1, 1,  ..., 0, 0, 0]])

 # [Konzolovvý výstup]: Segmentní maska velikosti @MAX_LEN:
  tensor([[4, 4, 4,  ..., 1, 1, 2],
          [4, 4, 4,  ..., 1, 1, 2],
          [4, 4, 4,  ..., 1, 1, 2],
          ...,
          [0, 0, 0,  ..., 1, 1, 2],
          [4, 4, 4,  ..., 1, 1, 2],
          [4, 4, 4,  ..., 1, 1, 2]])
```

Obr. 3.33: Výpis trénovacích tenzorů před napojením na neuronovou síť `XLNet`

- `NUMBER_OF_SENTENCES = 2` – počet `XLNet` vět během zpracování dat a procesu učení sítě (*maximálně povolené množství je 2* [52]);
- `TRAIN_DATA_SIZE = 30000` – množství entit použitých na trénování sítě;
- `VALID_DATA_SIZE = 10000` – velikost datového rámce při validaci sítě;
- `TEST_DATA_SIZE = 10000` – počet e-mailů použitých na testování sítě.

První parametr `LEARNING_RATE` byl v prvních fázích učení nastaven podle doporučených hodnot [52, 55]. Problém avšak spočíval v tom, že při jeho aplikování na datovou sadu e-mailů docházelo k přeučení sítě a ke klesnutí finální přesnosti modelu při aplikování na validační datovou sadu. V první řadě bylo sníženo množství epoch trénování při zachování doporučené rychlosti učení neuronové sítě. Výsledky daného učení jsou uvedeny v tabulce číslo 3.11. Je vidět, že obdržený výsledek s přesností v 99,53 % je velmi blízký k takovému u sítě `BERT`. Jedinou zápornou stránkou vytvořeného modelu bylo to, že při provedení následujících etap přeneseného učení docházelo ke snížení přesnosti a zhoršení výsledků `Loss` funkce. Dalším krokem k vyřešení daného problému bylo zúžení množiny trénovacích vzorků do 20 000 a snížení hodnoty *learning rate* do `LEARNING_RATE = 2e-5`.

Tab. 3.11: Výsledek trénování `XLNet` po jedné epoše trénování

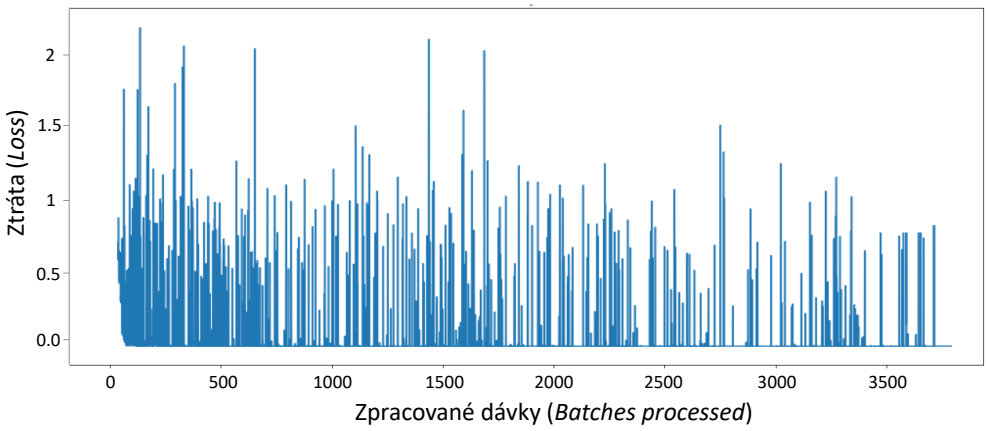
Epocha (*Epoch*)	Ztrátovost (*loss*)	Validační přesnost (*Validation accuracy*)	Čas (*Time*)
0	0,0643	**0,9953**	78 min 42 s

Snížení trénovacích a validačních vzorků negativně ovlivnilo výsledky učení a finální pravděpodobnost klesla o 4-5 %. Snížení rychlosti učení na hodnotu `2e-5` zapůsobilo na proces učení sítě pozitivně. Přestože síť po první epoše nedosahovala tak vysokých výsledků, bylo vidět že k přetrénování nedochází a je možné pokračovat v procesu přeneseného učení. Z takového důvodu byla zachována snížená hodnota rychlosti učení a provedeny dvě plné epochy trénování sítě `XLNet` na kompletní datové sadě elektronických zpráv. Výsledky dvou provedených etap přeneseného učení je možné vidět v tabulce číslo 3.12. Je z ní patrno, že po první epoše trénování, jejíž provedení zabralo 77 minut a 27 vteřin, finální přesnost klasifikátoru byla 98,96 % Trénovací ztrátovost se přitom pohybovala na úrovni 0,0725. Druhá epocha zabrala o 5 vteřin víc než první a celková doba učení trvala 77 minut a 33 vteřiny. Trénovací ztrátovost modelu byla 0,0220 a finální přesnost se rovnala 99,62 %. Absolutní přírůst přesnosti oproti první epoše přeneseného učení byl 0,66 %.

Tab. 3.12: Vyhodnocení procesu učení modelu sítě `XLNet`

Epocha (*Epoch*)	Ztrátovost (*loss*)	Validační přesnost (*Validation accuracy*)	Čas (*Time*)
0	0,0725	**0,9896**	77 min 27 s
1	0,0220	**0,9962**	77 min 33 s

Na obrázku číslo 3.34 lze vidět hodnotu `Loss` funkce během poslední etapy trénování. Z ní je patrné, že rozptyl hodnoty `Loss` byl vyšší, než u `BERT` a ve své maximální hodnotě převyšoval dva (u sítě `BERT` se hodnota `Loss` funkce pohybovala od 1,4). Pozorný čtenář si všimne, že finální výsledek `XLNet` je o 0,04 % horší než u modelu `BERT`, jehož přesnost se rovnala 99,66 %. Dosažení vyšší přesnosti na stejných validačních datech nebylo u modelu `XLNet` zaznamenáno. Z oficiálních statistik zmíněných v podkapitole číslo 3.3.3 je názorně vidět, že síť `XLNet` v případě provedení textové klasifikace dosahuje menší chybovosti, což by mělo mít vliv také na hodnotu finální přesnosti modelu. Odůvodnění obdrženého výsledku není na první pohled patrné, ale je s velkou pravděpodobností možné, a to z několika důvodů.



Obr. 3.34: Průběh `Loss` funkce u `XLNet` klasifikátoru

V prvé řadě za zmínku stojí skutečnost, že uvedené srovnání sítě `XLNet` oproti `BERT` na úlohu klasifikace dat bylo provedeno na modelech typu `Large`, které jsou naučené na odlišných datových sadách oproti modelům typu `Base`. Navíc objem datové sady modelu `XLNet-Base` byl podobný jako u příslušné varianty modelu `BERT`, viz [56]. Jako další faktor nižší přesnosti modelu `XLNet` je možné zdůraznit to, že datové sady spamových zpráv mají vlastní specifika a ve zveřejněných oficiálních statistikách [56] nebyly na úlohu klasifikace spamu otestovány. Ze strany autora této diplomové práce bylo věnováno maximální úsilí na zajištění co nejpodobnějších podmínek během testování, a to při minimálním ovlivnění ze strany různorodosti použitých dat a jiných trénovacích podmínek.

V tomto okamžiku lze tvrdit, že kapitola je u svého logického konce. V rámci ní došlo k úspěšnému trénování tří druhů moderních neuronových sítí na vyřešení problému rozpoznání nevyžádaného obsahu v elektronických zprávách. V prvé řadě, byly všechny tři modely podrobně nastudovány a byla popsána jejich specifika. V dalším kroku došlo k finálnímu přizpůsobení datové sady a provedení přeneseného učení sítí. Obdržené výsledky byly přehledně zobrazeny pomocí použitých knihoven na vykreslování průběhu funkcí a také pomocí kontroly takových parametrů jako je přesnost, doba trvání a trénovací ztrátovost. Ve finále daný přístup umožnil lépe popsat princip trénování modelů a pochopit jejich specifika. V následující kapitole budou natrénované modely podrobně porovnány na testovací datové sadě, která byla vybraná takovým způsobem, aby pokrývala co nejvyšší spektrum elektronických zpráv a co nejvíce přibližovala testovací podmínky k reálným, a to z hlediska dlouhodobého nasazení v rámci bezpečnostních mechanismů spamových filtrů.

4 Výsledky měření

V rámci předchozí kapitoly (viz kapitola číslo 3) byly úspěšně zprovozněny nejmodernější modely neuronových sítí přímo zaměřené na vyřešení klasifikační úlohy typu NLP spočívající v detekování nedůvěryhodných e-mailových zpráv na základě jejich textového kontextu. K otestování byly vybrány tři druhy hlubokých neuronových sítí. První z nich je model `ULMFiT`, který udělal revoluci ve světě NLP, a to jak svojí vysokou přesností a generalizací, tak i možností aplikování technik přeneseného učení během přizpůsobení na doménu konkrétní NLP úlohy, detailněji viz podkapitola 3.3.1. Druhým vybraným modelem byl `Google BERT` realizující myšlenku použití mechanismů pozornosti (*attention-based*) a hlubokou transformer architekturu. Vznik zmíněné architektury se stal v posledních dvou letech klíčovou událostí ve smyslu úrovně přesnosti modelů sítí a přesunul pozornost vědecké společnosti na danou technologii jako nejslibnější pro dosažení *state-of-the-art* výsledků v odvětví NLP. Samotná síť `BERT` posloužila jako motivace a důvod pro posunutí evoluční spirály předtrénovaných modelů transformer sítí a umožnila vznik světově známých modelů `Transformer-XL`, `ERNIE`, `RoBERTa`, `XLNet` a dalších. Více detailů je možné o dané problematice získat v podkapitole 3.3.2. Posledním vybraným modelem byla síť `XLNet`, která je stejně jako `BERT` postavena na architektuře transformerů, avšak realizující revoluční permutační techniky pro přesnější odhalení sémantických vazeb v textu. Daná síť byla předtrénována na největší datové sadě ze tří modelů a její předtrénování zabralo nejvíce GPU času, detailněji viz 3.3.3. Charakteristiky a hlavní výhody všech modelů byly podrobně rozebrány v kapitole číslo 3. Finální srovnání klíčových vlastností modelů je uvedeno v této kapitole v tabulce číslo 4.1.

Daná kapitola, kromě přímého srovnání modelu klade důraz na provedení testování vytvořených klasifikátorů na základě množiny e-mailů, která byla předpřipravena výhradně s motivací provedení testování a nebyla použita během procesu učení neuronových sítí. Po provedeném testování budou pro každý model vytvořeny matice záměn, spočítané celkové množství epoch přeneseného učení a určena doba jejich trvání. V dalším kroku bude na základě obdržených výsledků určena finální přesnost klasifikátorů na testovací datové sadě.

Navíc analýza současného stavu vědy a techniky provedená v rámci podkapitoly číslo 2.5 posloužila jako dobrý podklad pro vytvoření základní křivky úspěšnosti a srovnání modelů naučených v rámci diplomové práce se zveřejněnými výsledky ve světě umělé inteligence při aplikování na problém detekování nevyžádaného obsahu. Po provedeném srovnání bude možné určit finální kvalitu technik pro vytvoření datové sady a vhodnost vybraných DNN modelů. Na konci této kapitoly budou stručně definovány problémy vzniklé během implementační fáze projektu a budou nastíněna možná rozšíření a vylepšení do budoucna.

4.1 Postup testování

Testování modelů `BERT`, `ULMFiT` a `XLNet` bylo rozděleno do následujících kroků:

1. Načtení uložených modelů neuronových sítí;
2. Načtení testovací datové sady a její převod do vhodné podoby;
3. Definování sledovaných parametrů pro vyhodnocení úspěšnosti;
4. Provedení testování modelů na společné datové sadě;
5. Zpracování obdržených výsledků a srovnání se stavem vědy a techniky.

V prvním kroku došlo k opakovanému načtení všech tří natrénovaných DNN modelů pomocí technik přeneseného učení. Pro zpřehlednění klíčových vlastností každého modelu a efektivnějšího přímého srovnání byla vytvořena tabulka číslo 4.1.

Tab. 4.1: Finální srovnání charakteristik naučených modelů neuronových sítí

Vlastnost	`ULMFiT` [41]	`BERT` [43]	`XLNet` [51]
Základní architektura	AWD-LSTM	Bidirectional transformer	Permutation transformer
Počet vrstev jádra	3 LSTM vrstvy	12 kodér vrstev	12 kodér vrstev
Typ sítě	GLMP	GLMP	GLMP
Množství parametrů	$<$ 100 milionů[1]	110 milionů	110 milionů
Jazykový model	AWD-LSTM LM	AELM	ARLM
Tokenizace	MultiBatchEncoder	WordPiece	SentencePiece
Techniky	TWE	MLM, NSP	PLM, TSSA
Dimenzionalita vnoření	400	768	768
Optimizátor	Adam	Adam	Adam
Velikost bloku	–	512	512
Předtrénování LM (*Pre-training*)	Bez učitele (*unsupervised*)	Bez učitele (*unsupervised*)	Bez učitele (*unsupervised*)
Přenesené učení (*Fine-tunning*)	S učitelem (*supervised*)	S učitelem (*supervised*)	S učitelem (*supervised*)
Základní datová sada	`Wikitext-103`	`English Wikipedia`, `Books Corpus`	`BERT Datasets`, `Giga 5`, `ClueWeb`, `Common Crawl`
Doba trvání předtrénování LM[2]	1 GPU den	450 GPU dnů	2000 GPU dnů
Knihovna	`Fast.ai`	`PyTorch`	`PyTorch`

[2]Přesné množství parametru sítě se liší od finálního druhu NLP úlohy. Přibližný odhad byl vytvořen na základě informací dostupných z článku: <`https://arxiv.org/pdf/1609.07843.pdf`>.

[2]Je velmi problematické sjednotit výkony výpočetních zařízení, která byla použita pro předtrénování vybraných modelů. Jedná se o přibližné výsledky převedené na výkon jedné grafické karty za celý den své činnosti. Informace je dostupná z: <`https://www.borealisai.com/en/blog/understanding-xlnet`>.

Pro úspěšné načtení byly použity vestavěné nástroje zprostředkovávané knihovnami `fast.ai` a `PyTorch`. V rámci daného kroku byly použity optimalizované váhy sítí, vytvořená vnoření, architektury uložených modelů, konfigurační soubory apod. V textu dole bude zmíněná tabulka stručně popsána. Detailní charakteristiku modelů je možné přečíst v příslušných kapitolách 3.3.1, 3.3.2 a 3.3.3, anebo z oficiální dokumentace: `ULMFiT` [41], `BERT` [43] a `XLNet` [51].

Model `ULMFiT`, v jehož jádru leží tři vrstvy typu AWD-LSTM, představuje sám o sobě síť realizující GLMP (*General Language Model Pre-trained*). Celkové množství optimalizačních parametrů není v oficiálních dokumentech uvedeno a liší se na základě finální NLP úlohy. Vytvoření slovního vnoření probíhá v bloku MultiBatchEncoder a každé tokenizované slovo je vyjádřeno vektorem v čtyřsté dimenzi. Jako optimalizátor při procesu učení sítě byl použit Adam algoritmus. Předtrénování generalizovaného jazykového modelu probíhalo na datové sadě `Wikitext-103` technikou učení bez učitele[3] [62, 65]. Přenesené učení na datové sadě e-mailů bylo realizováno pomocí techniky učení s učitelem s přidáním e-mailových značek. Model `ULMFiT` použitý v této práci je dostupný skrz knihovnu `fast.ai`.

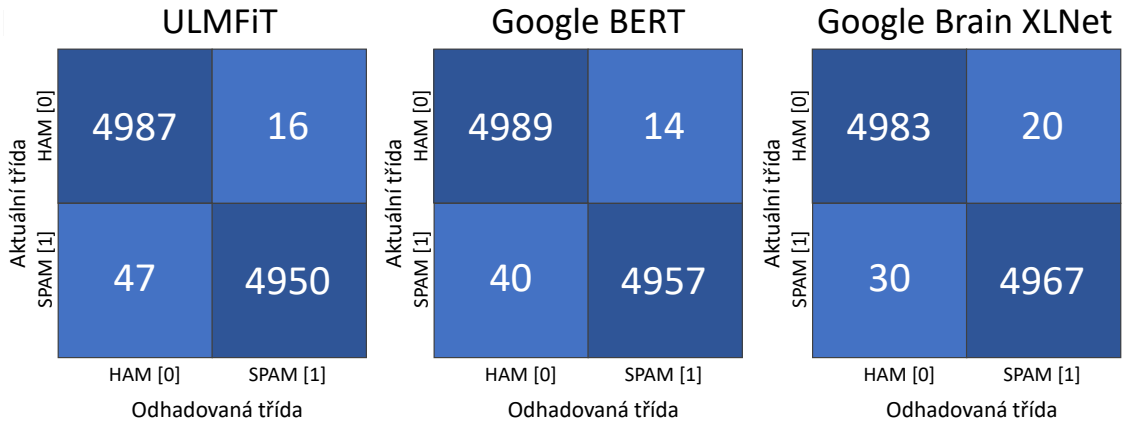
Model `BERT` představuje sám o sobě síť, v jejímž základu leží *bidirectional transformer* architektura. Jádro sítě je tvořeno dvanácti transformer kodér vrstvami a obsahuje celkem 110 milionů optimalizačních parametrů ve verzi `BERT-Base`. Stejně jako `ULMFiT` poskytuje `BERT` generalizovaný jazykový model GLMP typu AELM (*Autoencoder Language Model*). Tokenizace slov se provádí pomocí techniky WordPiece. Maximální délka jednoho bloku je omezena 512 tokeny a každé tokenizované slovo je zastoupeno vektorem v 768. dimenzi. Vnoření je docíleno technikou TWE (*Transferring Word Embeddings*). Pro lepší odhalení sémantických závislostí `BERT` realizuje techniky MLM (*Masked Language Modeling*) a NSP (*Next Sentence Prediction*). Pro optimalizaci parametrů je použit Adam mechanismus. Trénování původního modelu bylo provedeno na rozsáhlé datové sadě a trvalo v přibližném přepočtu 450 GPU dnů. Model je dostupný skrz knihovnu `PyTorch`.

Model `XLNet` realizuje architekturu transformerů, a vylepšuje ji pomocí permutačních technik. Jádro sítě se skládá z 12 kodér vrstev a celkové množství parametrů pro `XLNet-Base` činí 110 milionů. `XLNet` poskytuje generalizovaný jazykový model GLMP typu ARLM (*Autoregressive language model*). Tokenizace vstupní textové sekvence je docílena použitím knihovny SentencePiece. Dimenzionalita a maximální velikost bloků je totožná s odpovídajícími parametry modelu `BERT`. Navíc síť realizuje techniky PLM (*Permutation Language Modeling*) a TSSA (*Two-Stream Self-Attention*) k docílení lepších výsledků a překročení přesnosti `BERT`. Model je naučen na rozsáhlejší datové sadě než `BERT` a jeho trénování trvalo 2000 GPU dnů.

[3]Popis a srovnání jazykových modelů `ULMFiT`, `BERT` a `XLNet` je dostupný z URL: <`https://lilianweng.github.io/lil-log/2019/01/31/generalized-language-models.html`>.

Po úspěšném načtení tří modelů bylo možné přejít k dalšímu kroku a načíst testovací datovou sadu. Dá se říci, že testovací datová sada tvoří ve světě strojového učení jakýsi zlatý standart vyhodnocení přesnosti modelu a umožňuje provést jeho objektivní posouzení nezávislé na konkrétní datové sadě [17]. Posouzení přesnosti není možné provádět na validační datové sadě, jelikož byla bezprostředně použita při ladění systému, a proto v takovém případě dojde k částečnému vyvrácení podmínky nezávislosti [17, 67]. Samotná testovací datová sada byla vytvořena v rámci přípravy datových sad (viz podkapitola číslo 3.2.5). K procesu výběru dat bylo přistoupeno s co nejvyšší precizností, a především s motivací vytvořit ji takovým způsobem, aby elektronické zprávy pokrývaly co nejdelší časové období a plně odrážely dynamičnost vývoje spamových technik. V daném okamžiku je dobré zmínit, že po provedené transformaci dat, se testovací datová sada skládala z 10 000 náhodně zvolených e-mailových entit. Po náhodném výběru byla legitimní množina zastoupena 5003 e-maily. Datový rámec nevyžádané elektronické pošty obsahoval data za posledních 13 let (období roku 2006–2019), přičemž výběr byl udělán takovým způsobem, aby každý rok byl zastoupen náhodně vybraným blokem e-mailů pevně dané velikostí. Po znáhodněném formování testovacích dat byla množina e-mailů se spamovou značkou zastoupena 4997 entitami. Po úspěšném načtení byla testovací data převedená na podobu vyžadovanou každým druhem neuronové sítě a následně tokenizována, numerikalizována a napojena na vstupní vrstvy sítí.

V dalším kroku došlo k určení sledovaných parametrů, které umožní posoudit kvalitu naučených modelů. Pro objektivní srovnání výsledků byla do seznamu sledovaných parametrů kromě finální přesnosti začleněna matice záměn (*confusion matrix*). Jedná se o kontingenční matici, která se často používá při provádění statistických testů a posouzení klasifikačních schopnosti během procesů strojového učení [62]. Vytvořené matice záměn pro jednotlivé modely jsou uvedeny na obrázku číslo 4.1.



Obr. 4.1: Matice záměn úspěšnosti neuronových sítí `ULMFiT`,`BERT` a `XLNet`

Jak bylo zmíněno výše, matice byla vygenerována po ohodnocení 10 000 testovací e-mailových entit. V porovnání s klasickým pravděpodobnostním odhadem je matice záměn více vypovídající, poněvadž rozlišuje všechny možné dopady procesu klasifikace [62]. Jelikož se jedná o binární klasifikaci, počet možných stavů při provedení vyhodnocení odpovídá počtu buněk matice a je roven čtyřem. Jednotlivé buňky zastupují četnost toho, kolikrát během procesu testování vyskytla kombinace skutečné a predikované hodnoty. Četnosti na hlavní diagonále matice (označené tmavě modrou barvou) zastupují správně klasifikované případy, tj. skutečně pozitivní (*true positives*) a skutečně negativní (*true negatives*). Zbyle dvě buňky (označené světle modrou barvou) uvádí četnost chybných klasifikací, tj. falešně pozitivní (*false positives*) a falešně negativní (*false negatives*). Osy matice zastupují třídy odhadnuté klasifikátorem. Vertikální osa matice uvádí skutečné třídy důvěryhodných a nedůvěryhodných zpráv (značky tříd byly přidány během etapy transformace) a horizontální osa zastupuje odhadované třídy spamu a hamu modelem klasifikátoru.

První matice záměn (viz obrázek číslo 4.1) byla vytvořena při testování modelu `ULMFiT`. Druhá odráží výsledky sítě `Google BERT`. Poslední byla vytvořena po zprovoznění `XLNet`. Je patrné, že všechny tři klasifikátory ukázaly dobré výsledky při zpracování jim neznámé testovací sady. Model `ULMFiT` udělal pouze 63 neúspěšných pokusů. Celkový počet skutečně pozitivních odhadů, tj. situaci, kdy byla nevyžádaná zpráva klasifikována jako spam, byl roven 4 950. Počet správně klasifikovaných důvěryhodných zpráv, tj. skutečně negativních byl roven 4 987. Množství falešně pozitivních rozhodnutí (*false positives*) během nichž byla důvěryhodná zpráva klasifikována jako spam se rovnalo 16 a hodnota falešně negativních výsledků (spam byl propuštěn skrz bezpečnostní mechanismus) byla 47. Model `BERT` klasifikaci provedl úspěšněji. Celkové množství správných výsledků klasifikace se rovnalo 9 946, číslo falešných odhadů bylo 54. Ze skupiny správných výsledků bylo klasifikováno 4 989 legitimních e-mailů (skutečně negativní) a zastaveno 4 957 spamových zpráv (skutečně pozitivní). Množina falešně pozitivních výsledků obsahovala 14 zpráv a množina neodhalených spamů (*false negatives*) zahrnovala 40 entit. Model `XLNet` udělal neméně chybných hodnocení, jejichž množství bylo 50. Počet správných hodnocení se rovnal 9 950. Množství skutečně pozitivních případů, tj. správně vyhodnocených spamových zpráv se rovnal 4 967. Zbytek správných klasifikací byl roven 4 983. Chybná rozhodnutí byla rozdělena následujícím způsobem: 20 případů patřilo do skupiny falešně pozitivních a 30 falešně negativních. Zajímavou vlastností posledního modelu je to, že častěji než ostatní modely vyhodnocoval legitimní zprávy jako spam (skupina falešně pozitivních výsledků), ale zároveň udělal o 17 chyb méně[4] během přiražení spamovým zprávám značky „ham“ (skupina falešně negativních hodnocení).

[4]Ve srovnání s modelem `ULMFiT`.

Pokud se podrobněji zaměříme na obdržená čísla, tak dokážeme spočítat finální přesnosti testovaných modelů (viz tabulka číslo 4.2). První model v rámci testování, tj. `ULMFiT`, byl natrénován za sedm epoch přeneseného učení, přičemž tři z nich byly věnovány na vytvoření přizpůsobeného jazykového modelu a zbylých čtyři na samotné vytvoření klasifikátoru. Celková doba učení zabrala 66 minut GPU času[5].

Tab. 4.2: Výsledky testování vytvořených modelů neuronových sítí

Název modelu neuronové sítě	Epoch trénování	Doba učení	Správně ($TP + TN$)	Chybně ($FP + FN$)	Finální přesnost
`ULMFiT`	3 + 4	66 min	9 937 (−)	63 (−)	**99,37 %**
`Google BERT`	2	78 min	9 946 (+9)	54 (-14,29 %)	**99,46 %**
`Google Brain XLNet`	2	155 min	9 950 (+13)	50 (-20,63 %)	**99,50 %**

Finální přesnost `ULMFiT` na testovacích datech byla 99,37 %. Změna oproti vyhodnocení na validační datové sadě (podkapitola číslo 3.4.3) byla $99,44 - 99,37 = 0,07\,\%$. V tabulce byl model `ULMFiT` použit jako základ, a proto budou ve sloupcích hodnocení správně a chybně klasifikovaných e-mailů u modelů `BERT` a `XLNet` uvedena zlepšení přesnosti, anebo snížení chybných hodnocení oproti základu v podobě `ULMFiT`.

Model `BERT` byl naučen za dvě epochy trénování, které dohromady zabraly 78 minut GPU času. Změna oproti vyhodnocení na validačním rámci (viz podkapitola číslo 3.5.2) byla $99,66 - 99,46 = 0,2\,\%$. Oproti modelu `ULMFiT`, vyhodnotil `BERT` správně o devět e-mailů více a celkové množství udělaných chyb se snížilo o 14,29 %. Ačkoliv byla finální přesnost o $99,46 - 99,37 = 0,09\,\%$ vyšší než u `ULMFiT`, pokles přesnosti na testovací datové sadě byl u `BERT` vyšší a rovnal se $0,20 - 0,07 = 0,13\,\%$.

Poslední otestovaný model `XLNet` byl stejně jako `BERT` trénován dvě epochy. Celková doba učení byla ze všech modelů nejvyšší a rovnala se 155 minutám, což je skoro dvakrát více něž u sítě `BERT`. Přesnost modelu byla ze všech tři modelů nejvyšší a rovnala se $99,50\,\%$. Celkem zajímavou vlastností daného modelu bylo to, že po dokončení fáze trénování (viz podkapitola 3.6.2) se nacházel na druhé pozici, a to přímo za sítí `BERT`. Obsazení prvního místa během testování bylo docíleno menším absolutním poklesem přesnosti oproti výsledkům na trénovacích datech. Změna se rovnala $99,62 - 99,50 = 0,13\,\%$. Pro srovnání u `BERT` tento parametr byl roven $0,2\,\%$. Pokud `XLNet` srovnáme přímo s výsledky hodnocení `ULMFiT`, dospějeme k závěru, že neuronová síť `XLNet` odhadla o třináct e-mailových zpráv více a procentuální změna chybně klasifikovaných e-mailů se oproti `ULMFiT` snížila o $20,63\,\%$.

[5]Testování modelů bylo realizováno na hardware, již použitém během trénování, tj. grafickou kartou `Nvidia Xp` s podporou architektury CUDA a kapacitou paměti 12 196 MiB typu GDDR5X.

V následujícím kroku bylo provedeno hodnocení jiných sledovatelných parametrů, které bylo možné spočítat z matic záměn již popsaných výše. Konkrétně se jednalo o hodnoty preciznosti (*precision*), senzitivity (*recall*), specificity (*specificity*) a F-míru (*F-score*). Hodnota preciznosti vyjadřuje závislost mezi TP a součtem FP a TP. Laicky řečeno vyjadřuje to, jak moc se dá věřit skutečně pozitivnímu (TP) rozhodnutí klasifikátoru. Počítá se dle vztahu: $Preciznost = TP/(TP + FP)$. Hodnota senzitivity vyjadřuje závislost skutečně pozitivních rozhodnutí ke součtu falešně negativních a skutečně pozitivních rozhodnutí. Zjednodušeně řečeno vyjadřuje to, jaký podíl všech TP rozhodnutí bude klasifikátorem odhadnut. Počítá se dle vztahu: $Senzitivita = TP/(FN + TP)$, kde FN zastupuje množství spamových zpráv, které byly klasifikovány jako ham. Hodnota specificity, neboli *specificity* popisuje závislost skutečně negativních (TN) rozhodnutí k součtu falešně pozitivních (FP) a skutečně negativních (TN) výroků klasifikátoru. Počítá se dle vztahu: $Specificita = TN/(FP + TN)$. Posledním a jedním z nejpopulárnějších parametrů pro vyhodnocení klasifikátorů je F-míra. Daný parametr v sobě zahrnuje hodnoty preciznosti a senzitivity a počítá se dle vztahu: $\textit{F-míra} = 2 \cdot \frac{preciznost \cdot senzitivita}{preciznost + senzitivita}$. V tabulce číslo 4.3 jsou pro každý model výše zmíněné parametry spočítány.

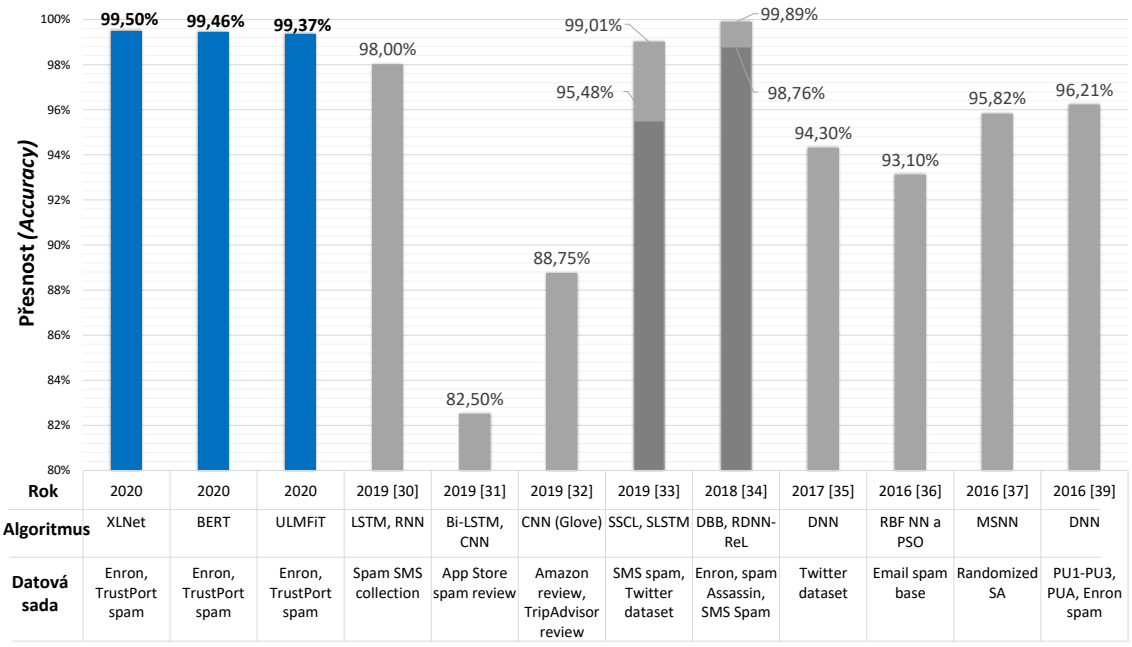
Tab. 4.3: Hodnocení klasifikátorů pomocí jiných stanovených parametrů

Název modelu neuronové sítě	Preciznost (*Precision*)	Senzitivita (*Recall*)	Specificita (*Specificity*)	F-míra (*F-score*)
`ULMFiT`	0,9968	0,9906	0,9968	0,9937
`Google BERT`	0,9972	0,9920	0,9972	0,9946
`Google Brain XLNet`	0,9960	0,9940	0,9960	0,9950

Z tabulky je patrné, že nejpreciznějším byl `BERT`, jehož hodnota se rovnala 0,9972. Druhou pozici obsadil `ULMFiT`, který dosáhl hodnoty 0,9968. Nejhůř dopadl model s nejvyšší finální přesností `XLNet`, jehož preciznost se rovnala 0,9960. Daný výsledek lze odůvodnit tím, že udělal ze všech tří modelů nejvíce falešně pozitivních (FP) rozhodnutí a označil ham jako spam. Hodnota senzitivity se rovnala: 0,9906 (`ULMFiT`), 0,9920 (`BERT`) a 0,9940 (`XLNet`). Takové výsledky se dalo očekávat, jelikož model `XLNet` udělal nejméně chyb při klasifikaci reálného spamu jako ham, tzn. hodnota falešně negativních rozhodnutí byla nejnižší. Hodnoty specificity vyšly pro všechny modely totožné s příslušnými hodnotami preciznosti. Hodnota F-míry (*F-score*) se pro modely rovnala: 0,9937 (`ULMFiT`), 0,9946 (`BERT`) a 0,9950 (`XLNet`).

Na konci této podkapitoly bylo rozhodnuto provést srovnání naučených modelů s existujícími mechanismy filtrování spamu zjištěnými v rámci provedené rešerše v podkapitole číslo 2.5. V rámci podkapitoly byly do tabulky číslo 2.1 vypsány

všechny veřejně dostupné vědecké výsledky realizující moderní algoritmy hlubokého učení za účelem provedení analýzy textového kontextu s jeho následující klasifikací. V daném okamžiku je důležité zmínit, že při srovnání těchto algoritmů strojového učení s modely neuronových sítí vytvořených v rámci diplomové práce, nelze opomenout skutečnost, že je velice problematické najít podobné modely hlubokých sítí, jejichž učení bylo provedeno na stejné datové sadě jako v daném experimentu. Z takového důvodu je nutno mít na paměti, že srovnání uvedené na obrázku číslo 4.2 je pouze orientační. Primárním cílem srovnání není provedení přímého porovnání a vyhodnocení přesnosti, ale ukázaní vhodnosti aplikování konkrétních architektur neuronových sítí na vyřešení zmíněného NLP problému a vytvoření kompletního přehledu. V případě, kdy bychom chtěli realizovat objektivní srovnání, bychom museli dodržet stejný postup, jaký byl dodržen i při porovnání vybraných modelů `ULMFiT`, `BERT` a `XLNet` a který spočíval v zajištění totožných trénovacích, validačních a testovacích datových rámců, a to za účelem vytvoření objektivního posouzení, které potlačí vlivy způsobené vstupními daty a umožní docílit stavu, kdy se budou finální výsledky modelů lišit pouze na implementovaných algoritmech strojového učení.



Obr. 4.2: Srovnání vytvořených modelů s existujícími modely klasifikátorů

Pokud se detailněji zaměříme na obrázek, zjistíme, že odráží informace uvedené v tabulce číslo 2.1. Jednotlivé vědecké práce a jejich maximální přesnosti jsou uvedeny v chronologickém pořádku. První tři sloupce (označeny tmavě modrou barvou) zastupují natrénované algoritmy `XLNet`, `BERT` a `ULMFiT`. Šedou barvou jsou označeny

jiné vědecké práce, jejichž výstupy umožňovaly je zahrnout do srovnání. V případě uvedení dvou dosažených přesností ve grafu (viz 4.2), bude se jednat o takovou práci (například [33, 34]), v níž bylo porovnáno několik algoritmů strojového učení. Přičemž do grafu následně budou zahrnuté takové výsledky, které splňují kriteria uvedena v kapitole 2.5. Horizontální rovina grafu zohledňuje klíčové informace podstatné pro porovnání. Jedná se o rok zveřejnění práce s odkazem na konkrétní článek, aplikovaný algoritmus a také datovou sadu, v jejímž základu byl model neuronové sítě natrénován. Detailní popis každé práce s uvedením autorů a klíčových specifik je možné vyčíst z kapitoly 2.5. Mezi silné stránky modelů diplomové práce lze zařadit jejich vysokou přesnost, univerzálnost a schopnost pracovat s jakýmkoliv e-mailovým kontextem po minimálním přizpůsobení dat bez nutnosti manuálního extrahovaní příznaků. Navíc v kombinaci se skripty vytvořenými v rámci práce, je možné proces vyhodnocení obdržené pošty plně automatizovat a docílit *end-to-end* zpracování. Za zmínku stojí, že nejpodobnějšími z hlediska použité veřejné datové sady `Enron` a účelů práce byly [34] a [39], které dosáhly přesnosti v 98,76 %, 99,89 % [34] a 96,21 % [39]. Ačkoliv se nejedná o objektivní srovnání modelů, z obrázku je také patrné, že jedna z neuronových sítí, která byla prezentována v [34], dosahuje výsledku 99,89 %, což je vyšší hodnota než u modelů natrénovaných v této diplomové práci. Konkrétně se jedná o modifikaci sítě DBB-RDNN-ReL při dvou skrytých vrstvách a 2000 přidaných vlastností, viz [34]. Prezentovaná síť není postavena na architektuře transformerů (oproti modelům `BERT` a `XLNet`) a je natrénována na odlišné datové sadě. Odlišnosti ve vstupních datech mohly posloužit jako klíčový bod vyšší finální přesnosti modelu DBB-RDNN-ReL. Zjištění dané každopádně skutečnosti poslouží jako dobrá motivace a vize do budoucna směřující k vylepšení prezentovaných technik a k ověření jejich přesnosti na stejné datové sadě jako v [34].

4.2 Problémy vzniklé během implementace

Není možné říci, že proces implementace všech výše zmíněných kroků proběhl bez komplikací. Vyřešení některých z nich bylo záležitostí několika minut, a některé zabrzdily proces realizace naplánovaných kroků na několik dnů. Ve finále lze každopádně konstatovat, že všechny problémy, které vznikly v rámci diplomové práce byly úspěšně vyřešeny. Dokončení všech implementačních kroků umožnilo provést naučení a srovnání tří modelů neuronových sítí, dosáhnout vysokých přesností a realizovat cíle definované v zadání diplomové práce.

Obecně je množinu všech problémů možné rozdělit na dvě základní podmnožiny. První zahrnuje problémy, jejichž vyřešení bylo nutné pro realizaci parsovacích skriptů použitých během přípravy datové sady a převodu e-mailů do přijatelné podoby.

Druhá množina byla zastoupena takovými problémy, které nastaly během finální přípravy datové sady a také při trénování modelů hlubokých neuronových sítí.

Nejpodstatnější problémy v první fázi implementace byly:

1. Finální sjednocení surových dat z různých zdrojů;
2. Předzpracování e-mailů, jejich dekódování a oprava chyb;
3. Problematické parsování záměrně poškozených e-mailových zpráv;
4. Odstranění stylistických částí textu a multimediálních příloh;
5. Vytvoření efektivních regulárních výrazů pro kontrolu a extrahování informací;
6. Finální transformace dat do třech datových rámců;
7. Efektivní uložení zpracovaných dat pro budoucí využití.

První z množiny těchto problémů vznikl při sběru datového rámce nedůvěryhodných e-mailů. Jelikož se jednalo o datovou sadu poskytnutou společností TrustPort, bylo nutné spamová data manuálně roztřídit dle roku, typu zprávy[6] a formátu uložení[7]. Druhý a třetí problém vznikl při přímém zpracování e-mailů z důvodu vytvoření útočníky záměrně poškozených zpráv s cílem obejití bezpečnostních mechanismů na straně příjemce, detailněji viz podkapitolu číslo 1.4. Konkrétně se jednalo o problémy spojené s dekódováním zpráv s nesprávně uvedeným kódováním hlavičky, použití netisknutelných speciálních symbolů, které nebyly správně rozpoznány na straně vytvořeného dekodéru atd. V čtvrtém kroku došlo k problému spojenému s odstraněním částí HTML a CSS, které byly do těla e-mailů vnořeny netradičním způsobem a nebyly viditelné ani pomocí externích `Python` knihoven. K eliminaci daného problému byly použity regulární výrazy v kombinaci s nástroji třetích stran. Navíc byly regulární výrazy aplikovány pro odstranění symbolů s nulovou informační hodnotou, pro extrahování URL odkazů atd. Poslední dva problémy byly spojeny s principem výběru a uložení trénovací, validační a testovací datové sady. Bylo nutné zvolit optimální velikosti datových rámců a naplnit je různorodými daty, aby co nejlépe pokryty dynamičnost vývoje spamových technik, viz 3.2.5.

Do druhé podmnožiny problémů bylo zařazeno:

1. Problémy s finálním přizpůsobením dat k modelování;
2. Přenos kódu z Google Colab na fyzický server `Gravy`;
3. Nekompatibilita nainstalovaných knihoven a driverů;
4. Potíže během tokenizace a numerikalizace textových sekvencí;
5. Chyby při vytvoření příslušných masek pro `BERT`, `XLNet`;
6. Přeplnění paměti, vznik chyby: `CUDA out of memory`;
7. Problémy s přeučením sítě `XLNet`.

[6]Bezpečnostní řešení společnosti TrustPort data třídila dle prezence multimediálních příloh v těle zprávy anebo existenci spustitelných souborů.

[7]Tím je myšlena skutečnost, že během svého vývoje byl produkt vylepšován a rozšiřován, v důsledku čehož docházelo ke změně logiky zpracování přijatých e-mailů a způsobu jejich uložení.

První problém z druhé fáze realizace praktické části této diplomové práce nastal hned během finálního přizpůsobení dat k modelování. Před tím, než bylo provedeno načtení uložených souborů, bylo nutné podrobně nastudovat princip fungování vybraných druhů neuronových sítí, nainstalovat a seznámit se s knihovnami `Pytorch` a `fast.ai`. Až po dokončení těchto kroků bylo možné přistoupit k vytvoření individuálních `Python` tříd sloužících pro převod datových sad do požadované podoby a načtení dat do paměti. V dalším kroku vznikl problém s přenosem vytvořeného zdrojového kódu z Google Colab na fyzický server `Gravy`, laskavě poskytnutým vedoucím této diplomové práce. Jak bylo zmíněno v implementačním procesu u každého modelu, základní inicializace a první spouštění trénování na redukovaných datech probíhalo v Google Colab. Následně, pokud byl daný implementační krok úspěšný, docházelo k přenosu naprogramovaného kódu a ke spouštění na reálném hardware. Základní problémy při přenosu kódu byly spojeny s verzemi nainstalovaných driverů a s knihovnami nutnými pro inicializaci neuronových sítí. Tento problém byl vyřešen vytvořením virtuálních `Python` prostředí s příslušnými knihovnami pro každý model sítě zvlášť, a to pomocí použití frameworku `Conda`. K následujícím problémům je možné zařadit tokenizaci a numerikalizaci textových sekvencí a také vytvoření maskovacích sekvencí pro dvě vstupní věty `BERT` a `XLNet`. Před začátkem učení neuronových sítí `BERT` a `XLNet` často vznikal problém `CUDA out of memory`, který byl vyřešen zmenšením entit v jedné dávce (proměnná *batch size*) a rozdělením vytvořených tensorů na podmnožiny pomocí externích nástrojů (viz zdrojový kód). V rámci učení sítě `XLNet` docházelo k častému přeučení sítě a nízkým klasifikačním výsledkům na validační datové sadě. Vyřešení problému viz podkapitola 3.6.2. Zpozdění finální testovací fáze bylo navíc způsobené opakovaným načtením natrénovaných sítí a implementací sledovatelných parametrů.

4.3 Rozšíření do budoucna

Po dokončení této diplomové práce není plánováno výzkumné činnosti v tomto směru zastavovat. V budoucnu bude provedeno otestování čtvrtého modelu neuronové sítě, v podobě modelu `GPT-2`, jehož zprovoznění a otestování v rámci této diplomové práce nebylo realizováno, a to z důvodu nedostačujících paměťových kapacit dostupné grafické karty a problémům s vystavěným modulem provádějícím tokenizaci textu. Jelikož praktické kroky této diplomové práce byly navrženy takovým způsobem, aby bylo možné docílit *end-to-end* procesu zpracování vstupních e-mailů, bude vybraná neuronová síť v budoucnu nasazena na reálně běžící Gateway server, kde budou ověřeny její klasifikační schopnosti během reálné elektronické komunikace. Ve finále je s vedoucím této diplomové práce plánováno zveřejnění výsledků provedeného výzkumu do impaktovaného vědeckého časopisu v angličtině.

Závěr

Závěrem je možné k napsané diplomové práce zmínit, že všechny úlohy definované v rámci zadání práce byly splněny. Kromě toho se proces tvorby po dosažení definovaných cílů práce a její praktické části nezastavil, ale plynule pokračoval dál, a to především s motivací dosáhnout co nejlepších výsledků a vytvořit takovou práci, která by z teoretického hlediska měla reálný vědecký přínos a z praktického úhlu pohledu poskytovala podrobné srovnání nejmodernějších druhů neuronových sítí dnešní doby pro vyřešení úlohy klasifikace nevyžádané pošty a tvořila funkční řešení, které by bylo reálně použitelné jak ve světě počítačové bezpečnosti, tak i v rychle se měnícím oboru umělé inteligence.

Teoretická část diplomové práce byla věnována problematice elektronické komunikace a filtrování e-mailových zpráv pomocí technik umělé inteligence, viz kapitoly číslo 1 a 2. V rámci první kapitoly důraz byl kladen na pochopení principů fungování elektronické komunikace, jejího zabezpečení a právní úpravu. Pochopení základních principů bylo nezbytné, a to jak při studiu odborné literatury, tak i pro implementaci bezpečnostních řešení spojených s danou problematikou. V první řadě byla pozornost věnovaná vytvoření globálního přehledu o rozšířenosti služeb elektronické komunikace, viz kapitola číslo 1. Následně byl vytvořen technický pohled na principy realizace e-mailové komunikace. V rámci této části byla popsána architektura e-mailové komunikace, její základní stavební prvky a také protokoly, jejichž použití umožňuje doručení odeslané zprávy adresátovi, viz podkapitola 1.1. Z důvodu zaměření této diplomové práce na filtrování nevyžádaných elektronických zpráv a detekci spamových útoků bylo nezbytné popsat formát a strukturu e-mailových zpráv, používaná kryptografická primitiva a existující e-mailová rozšíření. V rámci daného popisu byla zanalyzována struktura záhlaví a těla e-mailů, byl vytvořen seznam povinných a nepovinných polí hlavičky a také byly popsány způsoby a možnosti přenášení různorodého obsahu, viz podkapitola 1.2. V dalším kroku došlo k vytvoření popisu principů fungování spamu. Na základě statistik volně dostupných na internetu byl vytvořen globální přehled výskytu spamových zpráv v internetovém prostředí, viz podkapitola 1.3. Statistiky zahrnovaly informace o podílu spamu na celkovém objemu e-mailových zpráv, seznam nejpopulárnějších anglických spamových slov a seznam států, na jejichž území dochází ke generování nejvyššího množství spamového provozu atd. Z důvodu motivace k vytvoření co nejvíce komplexního přehledu o problematice elektronické komunikace, byla do teoretické části zahrnuta právní regulace spamových zpráv jak v rámci České republiky, tak i Evropské unie (EU). Na konci byl vytvořen technický pohled na evoluci spamových technik a principů obcházení bezpečnostních mechanismů, viz podkapitola 1.4. V rámci technického pohledu na problematiku spamu byly klasifikovány a detailně rozebrány způsoby

detekce a ochrany. Rozdělení bylo provedeno na tři skupiny: *reputation-based*, *textual content-based* a *multimedia content-based* (detailněji, viz podkapitola číslo 1.5).

V rámci druhé, teoreticky zaměřené kapitoly, byl vytvořen úvod do problematiky umělé inteligence a hlubokého učení, viz podkapitola číslo 2.1. V rámci úvodu byl proveden podrobný rozbor taxonomie strojového učení a hlubokých neuronových sítí. Byly popsány základní stavební prvky umělých neuronových sítí, existující architektury, jejich struktura a odlišnosti, viz podkapitola 2.2. V rámci další podkapitoly byly popsány moderní techniky učení hlubokých neuronových sítí (viz podkapitola 2.3), jejich klíčové vlastnosti, možné aplikování a optimalizace. Následně byla hlavní pozornost věnována problematice zpracování textu pomocí metod hlubokého učení a zjištění aktuálního stavu vědy a techniky v rámci oboru filtrování spamových zprav, viz podkapitola 2.4. Při popsání specifik zpracování textových sekvencí byl vyjmenován okruh NLP problémů, jehož vyřešení je velmi aktuálním pro odborníky ze světa textové analýzy a strojového učení. V rámci podkapitoly byly podrobně vysvětleny problémy při odhalení sémantických vazeb v textu a popsány existující architektury a techniky, které lze pro vyřešení NLP úloh aplikovat. Následně byl uveden popis procesu převodu textových sekvencí na tokeny, jejich numerikalizaci a zakódování do vícedimenzionálního vektorového prostoru pomocí techniky vnoření slov. Na konci podkapitoly byly stručně popsány a porovnány moderní neuronové sítě `ULMFiT`, `Google BERT`, `Open AI GPT-2` a `Google Brain XLNet`, které lze na zpracování textu úspěšně aplikovat, viz podkapitola 2.5. V závěru druhé části této diplomové práce byla pozornost věnována moderním technikám strojového učení, které se v moderních bezpečnostních řešeních na filtrování spamu používají. Každá z technik byla stručně rozepsána, viz podkapitola 2.5. Popis technik zahrnoval princip jejich fungování, existující algoritmy, které je adaptují, dosažitelné přesnosti a specifika použití. Důraz byl kladen na algoritmy hlubokého učení, během nějž byla provedena globální rešerše aktuálních vědeckých prací, které se zároveň zabývaly otázkami neuronových sítí a detekování nevyžádaného textového obsahu neboli spamu. Primární motivací rešerše v podkapitole číslo 2.5 bylo vytvoření podkladů pro zkonstruování základní křivky úspěšnosti algoritmů, pochopení tendencí vývoje bezpečnostních řešení, provedení srovnání prezentovaných algoritmů podle efektivity, finální přesnosti, použitých datových sad apod. Na konci druhé kapitoly byla vybrána taková vědecká studia, která jsou nejvíce srovnatelná s modely realizovanými v této diplomové práci, jejichž klíčové vlastnosti a dosažené přesnosti byly přehledně prezentovány ve finální tabulce číslo 2.1.

Prakticky zaměřené části byly úzce spojené s vytvořením funkčních modelů vybraných neuronových sítí, které by mohly provádět klasifikaci vstupních e-mailových zpráv, tj. provádět rozdělení zpráv do dvou kategorií na základě analýzy jejich kontextu, viz kapitola číslo 3. Základním prvkem procesu učení jakéhokoliv algoritmu

strojového učení jsou data. Čím kvalitnější je vstupní datová sada během učení modelu, tím kvalitnější výsledek lze očekávat. Z tohoto důvodu byl v počátečních praktických kapitolách kladen důraz na vytvoření detailního popisu dvou vstupních datových sad (spam a ham), a to s cílem lepšího pochopení jejich struktury, různorodosti dat, vyskytujících jazyků apod. Množina důvěryhodných e-mailů byla zastoupena volně dostupnou datovou sadou `Enron email dataset`, viz podkapitola 3.1.1. Tento datový rámec anglických e-mailů byl zveřejněn v roce 2015 a je zastoupen 517 401 entitami, které jsou uloženy ve formátu CSV. Množina nedůvěryhodných e-mailů byla ochotně poskytnuta společností TrustPort a obsahovala interní spamové e-maily nasbírané bezpečnostními prvky společnosti za posledních 13 let (rok 2006–2019), viz podkapitola 3.1.2. Datová sada byla zastoupena surovými e-maily v různých jazycích, jejichž celkové množství bylo rovno 720 034. V následujících krocích bylo provedeno zpracování datových sad a z nich vytvoření finální datové sady, která byla použita během procesu učení vybraných modelů. Fáze zpracování dat zahrnovala kroky základního předzpracování, čištění, výběru, integraci a transformaci dat, viz kapitola číslo 3.2.3. Primární motivací všech výše popsaných kroků byl převod e-mailových entit do takové podoby, aby byla co nejvíce přijatelná pro napojení na vstupy do neuronových sítí při zachování informační hodnoty každé zprávy. Během zpracování dat došlo k jejich jazykovému třídění, potlačení negativních vlivů záměrně vnesených do struktury spamových zpráv, odstranění nepodstatných vstupů pro klasifikační mechanismy – textových dat, multimediálních příloh a binárních souborů. Po provedení všech výše uvedených modifikací a zpracování, obsahovala vytvořená finální datová sada 1 070 186 anglických e-mailových entit, viz podkapitola 3.2.4. V posledním kroku zpracování dat došlo k inteligentnímu výběru a označkování potřebné množiny e-mailů (podkapitola číslo 3.2.5), a to s cílem dosažení maximální různorodosti vstupních dat a předejití přetrénování modelů neuronových sítí během aplikování přeneseného učení (*transfer learning*). Finální datová sada byla rozdělena na tři datové rámce sloužící pro trénování, testování a validaci budoucích modelů. Za zmínku také stojí skutečnost, že výše uvedené kroky byly udělány takovým způsobem, aby finální sada naprogramovaných skriptů na zpracování dat byla co nejvíce generalizována a v budoucnu bylo možné docílit end-to-end zpracování a vyhodnocení e-mailových entit v reálném čase.

Po provedení předzpracování dat bylo přistoupeno k realizaci druhé prakticky zaměřené etapy diplomové práce, viz podkapitola číslo 3.3. Daná etapa byla stěžejní a zahrnovala v sobě provedení výběru a natrénování hlubokých sítí na vyhodnocení spamového obsahu. V rámci této etapy byly vybrány tři nejmodernější architektury neuronových sítí: `ULMFiT`, `BERT` a `XLNet`. V textu práce byly popsány silné stránky zvolených modelů, jejich klíčové vlastnosti, princip fungování a architektura, viz podkapitoly číslo 3.3.1, 3.3.2 a 3.3.3. Logicky bylo možné proces vytváření funkčních

klasifikátorů, v jejichž základu leží natrénované modely neuronových sítí, rozdělit do tří klíčových etap. V první etapě došlo k finálnímu přizpůsobení dat k modelování, v rámci kterého bylo nutné zohlednit specifika každého modelu a transformovat datovou sadu do takové podoby, aby byla sítěmi úspěšně dekódována a zpracována. V druhé etapě, tj. po provedení přizpůsobení dat k modelování, byly vstupní textové sekvence tokenizovány (operace *tokenization*) a numerikalizovány (operace *numericalization*). Následně došlo k převodu numerikalizovaných tokenů do vícedimenzionálního vektorového prostoru, k jejich napojení na vstupní vrstvu inicializovaného modelu hluboké neuronové sítě a provedení procesu přeneseného učení. V případě učení modelu `ULMFiT` zahrnoval proces trénování taktéž přizpůsobení generalizovaného jazykového modelu, a to s cílem dosažení ještě lepších výsledků a dokonalejšímu pochopení principů přeneseného učení, viz podkapitola číslo 3.4. Po provedeném naučení jazykového modelu `ULMFiT` bylo uskutečněno jeho statistické testování s odhadem finální přesnosti, která se rovnala 60,08 %. Navíc v rámci trénování `ULMFiT` byla provedena interaktivní ukázka schopnosti naučeného modelu generovat text na základě vstupní věty s použitím slovníku e-mailových zpráv vygenerovaného v rámci předchozích epoch trénování, viz podkapitola 3.4.2. Po aplikování přeneseného učení byla architektura jazykového modelu `ULMFiT` upravena a připravena k trénování `ULMFiT` klasifikátoru. Přenesené učení dvou zbylých modelů hlubokých neuronových sítí (viz podkapitoly čísla 3.5 a 3.6), tj. `BERT` a `XLNet` probíhalo odlišným způsobem, a to z toho důvodu, že dané modely nerealizují `AWD-LSTM` síť, ale pokročilejší architekturu typu Transformer, v níž jsou na problém klasifikace nevyžádané pošty aplikovány různé mechanismy pozorností, detailněji o principech fungování a výhodách transformer sítí viz text kapitoly 3.3.2. Po přizpůsobení trénovací a validační datové sady a také jejich napojení na vstupní vrstvy neuronových sítí byly všechny tři modely natrénovány. Jednotlivé kroky přeneseného učení sítí byly v textu kapitoly detailně popsány stejně jako i proces určování takových parametrů jako je rychlost učení (*learning rate*), velikost dávky (*batch size*), počet epoch trénování a dalších. Tyto parametry byly pro každou síť nastaveny takovým způsobem, aby proces učení probíhal co nejefektivněji s maximálním využitím paměťových kapacit dostupného hardware. Během trénování byly sledovány hodnoty `Loss` funkce, hodnoty validační přesnosti učícího se klasifikátoru a trvání jednotlivých epoch optimalizace modelů. Po provedení všech epoch učení byly na základě validační datové sady spočítány finální přesnosti. Přesnost klasifikátoru `ULMFiT` byla 99,44 %, pro `BERT` se rovnala 99,66 % a pro `XLNet` 99,62 %.

V poslední kapitole věnované zhodnocení výsledků diplomové práce (viz kapitola 4) byly vytvořené klasifikátory otestovány v obdobných podmínkách jako jsou v reálném prostředí. Nezávislé testování bylo provedeno na speciálně vybrané testovací množině e-mailů, která nebyla zapojena do procesu ladění výše popsaných modelů.

Na začátku této kapitoly (viz podkapitola číslo 4.1) byl stručně popsán postup provedení testování a byly definovány sledované parametry pro vyhodnocení úspěšnosti modelů. V rámci daného postupu došlo k vytvoření závěrečné tabulky číslo 4.1 zahrnující finální charakteristiky naučených modelů. Následně byla uvedena stručná charakteristika odlišností existujících v modelech `ULMFiT`, `BERT` a `XLNet`. Dále se podkapitola číslo 4.1 věnuje popisu testovací datové sady. Do popisu bylo zahrnuto rozložení e-mailových entit a byl zmíněn postup vytvoření dané množiny. Pro objektivní posouzení výsledků byla do seznamu sledovaných parametrů, kromě finálních přesnosti, zahrnuta matice záměn (*confusion matrix*). Výsledky spamových klasifikátorů na testovacích datech byly následující: `ULMFiT` ($TN = 4987$; $TP = 4950$; $FN = 47$; $FP = 16$), `BERT` ($TN = 4989$; $TP = 4957$; $FN = 40$; $FP = 14$) a `XLNet` ($TN = 4983$; $TP = 4967$; $FN = 30$; $FP = 20$), viz obrázek číslo 4.1. Kompletní popis a vysvětlení každé klasifikační skupiny byl uveden přímo v textu příslušné kapitoly. Model `ULMFiT` byl natrénován za sedm epoch učení, které dohromady trvaly 66 minut. Finální přesnost byla u `ULMFiT` na testovacím datovém rámci rovna **99,37 %**. Hluboká síť `BERT` byla trénována během dvou epoch trénování, které trvaly 78 minut. Finální přesnost `BERT` byla **99,46 %**. Poslední model `XLNet` byl natrénován za 155 minut (dvě epochy učení) a jeho přesnost se rovnala **99,50 %**. Všechny výše uvedené hodnoty byly v textu podrobně okomentovány a uvedeny v tabulce číslo 4.2. Následně byly na základě obdržených matic záměn spočítány pro každou síť hodnoty (tabulka číslo 4.3) preciznosti (*precision*), senzitivity (*recall*), specificity (*specificity*) a F-míry. Navíc po provedeném testování byly vytvořené modely klasifikátorů porovnány s existujícími řešeními zjištěnými v rámci rešerše podkapitoly číslo 2.5. Ačkoliv byly srovnávané modely natrénovány na různých datových sadách nejedná se o přímé srovnání, ale pouze orientační, které má za cíl vytvořit základní křivku (*base-line*) a ukázat vhodnost aplikování moderních technik strojového učení na problém filtrování spamu. Na konci kapitoly 4 byly prodiskutovány problémy vzniklé během implementace (viz podkapitola číslo 4.2) a v jejím závěru detailně popsány kroky, které jsou plánované v budoucnu uskutečnit, viz podkapitola číslo 4.3.

Literatura

[1] KABELOVÁ, Alena a Libor DOSTÁLEK. *Velký průvodce protokoly TCP/IP a systémem DNS*. 5., aktualiz. vyd. Brno: Computer Press, 2008. ISBN 978-80-251-2236-5.

[2] JEŘÁBEK, Jan. *Komunikační technologie*. Brno: Vysoké učení technické v Brně, 2017. ISBN 978802144446.

[3] CROCKER, Stephen. *Internet Mail Architecture: RFC 5598* [online]. 2009 [cit. 3. 12. 2019]. Dostupné z: <`https://tools.ietf.org/html/rfc5598`>.

[4] Email Statistics Report: 2019-2023. *The Radicati Group, Inc.: Technology Market Research Firm* [online]. [cit. 29. 11. 2019]. Dostupné z: <`https://www.radicati.com`>.

[5] AVOINE, Gildas, Pascal JUNOD a Philippe OECHSLIN. *Computer System Security: Basic Concepts and Solved Exercises* [online]. EPFL Press, 2007 [cit. 3. 12. 2019]. ISBN 9781420046205. Dostupné z: <`https://books.google.cz/books?id=UwrOhcgVhsMC&source=gbs_navlinks_s`>

[6] VERGELIS, Maria, Tatyana SHCHERBAKOVA a Tatyana SIDORINA. *Spam and phishing in Q1 2019: Kaspersky Security* [online]. 2019 [cit. 10. 12. 2019]. Dostupné z: <`https://securelist.com/spam-and-phishing-in-q1-2019/90795/`>.

[7] VERGELIS, Maria, Tatyana SHCHERBAKOVA a Tatyana SIDORINA. *Spam and phishing in Q2 2019: Kaspersky Security* [online]. 2019 [cit. 10. 12. 2019]. Dostupné z: <`https://securelist.com/spam-and-phishing-in-q2-2019/92379/`>.

[8] REIFOVÁ, Irena. *Slovník mediální komunikace*. Praha: Portál, 2004. ISBN 80-7178-926-7.

[9] ŠILHAVÝ, Pavel. *Datová komunikace* [online]. Brno: Vysoké učení technické v Brně, 2012 [cit. 10. 12. 2019]. ISBN 9788021444553. Dostupné z: <`https://www.vutbr.cz/www_base/priloha.php?dpid=68492`>.

[10] BHOWMICK, Alexy a Shyamanta M HAZARIKA. E-Mail Spam Filtering: A Review of Techniques and Trends. *Advances in Electronics, Communication and Computing* [online]. 2018 [cit. 8. 12. 2019]. Dostupné z: <`https://arxiv.org/abs/1606.01042`>.

[11] DADA, Emmanuel Gbenga, Joseph Stephen BASSI a Haruna CHIROMA. Machine learning for email spamfiltering: review, approaches and openresearch problems. *Heliyon* [online]. 2019, 23 str. [cit. 20. 3. 2020]. Dostupné z: <https://reader.elsevier.com/reader/sd/pii/S2405844018353404>.

[12] *Výhody datových schránek: Datových schránky* [online]. 2018 [cit. 8. 12. 2019]. Dostupné z: <https://www.datoveschranky.info/o-datovych-schrankach/vyhody-datovych-schranek>.

[13] WOLFORD, Ben. How does the GDPR affect email? *Complete guide to GDPR compliance* [online]. [cit. 7. 12. 2019]. Dostupné z: <https://gdpr.eu/email-encryption/>.

[14] MILLER, Jeremy. *History: How SPAM got its name* [online]. 2018 [cit. 3. 12. 2019]. Dostupné z: <https://stickybranding.com/how-spam-got-its-name/>

[15] KEE, Jared. *Social Engineering: Manipulating the Source* [online]. SANS Institute, 2008, 32 s. [cit. 4. 12. 2019]. Dostupné z: <https://www.sans.org/reading-room/whitepapers/engineering/social-engineering-manipulating-source-32914>.

[16] TLUSŤÁK, Karel a Kamil MALINKA. *Problematika emailové komunikace: Email Communication Problematics*. Brno: Vysoké učení technické, Fakulta informačních technologií, 2008.

[17] BURGET, Radim. *Teoretická informatika*. Brno: Vysoké učení technické v Brně, 2013. ISBN 9788021448971.

[18] CHOLLET, Francois a Rudolf PECINOVSKÝ. *Deep learning v jazyku Python: Knihovny Keras, TensorFlow*. Grada, 2019. ISBN 9788024731001.

[19] CHOLLET, Francois. *Deep learning with Python*. Shelter Island, New York: Manning Publications Co., [2018]. ISBN 1617294438.

[20] SIEFKES, Christian a William YERAZUNIS. *Combining Winnow and Orthogonal Sparse Bigrams for Incremental Spam Filtering* [online]. 2003 [cit. 5. 4. 2020]. Dostupné z: <http://alumni.cs.ucr.edu/~schhabra/winnow-spam.pdf>.

[21] GARG, Rohit. *A PRIMER TO ENSEMBLE LEARNING — BAGGING AND BOOSTING* [online]. [cit. 5. 4. 2020]. Dostupné z: <https://analyticsindiamag.com/primer-ensemble-learning-bagging-boosting/>.

[22] XIN-SHE, Yang. *Firefly Algorithms for Multimodal Optimization* [online]. Springer Berlin Heidelberg, 2009 [cit. 6. 4. 2020]. ISSN 9783642049446. Dostupné z: <https://link.springer.com/chapter/10.1007\/978-3-642-04944-6_14>.

[23] WHISSELL, John a Charles CLARKE. *Clustering for semi-supervised spam filtering* [online]. [cit. 7. 4. 2020]. Dostupné z: <https://dl.acm.org/doi/pdf/10.1145/2030376.2030391>.

[24] KARIM, Masud a Rashedur RAHMAN. *Decision Tree and Naïve Bayes Algorithm for Classification and Generation of Actionable Knowledge for Direct Marketing* [online]. 2013 [cit. 7. 4. 2020]. Dostupné z: <https://www.scirp.org/pdf/jsea_2013042913162682.pdf>.

[25] LANDWEHR, Niels, Mark HALL a Eibe FRANK. *Logistic Model Trees* [online]. [cit. 7. 4. 2020]. Dostupné z: <https://www.cs.waikato.ac.nz/~ml/publications/2003/landwehr-etal.pdf>.

[26] KOHAVI, Ron. Scaling Up the Accuracy of Naive-Bayes Classifiers. *Aaai.org* [online]. [cit. 7. 4. 2020]. Dostupné z: <https://www.aaai.org/Papers/KDD/1996/KDD96-033.pdf>.

[27] BREIMAN, Leo, SCHAPIRE, Robert, ed. *Random Forests* [online]. 2001 [cit. 9. 4. 2020]. Dostupné z: <https://link.springer.com/content/pdf/10.1023/A:1010933404324.pdf>.

[28] ŠÍMA, Jiří a Roman NERUDA. *Teoretické otázky neuronových sítí*. Praha: MATFYZPRESS, 1996. ISBN 80-85863-18-9.

[29] MINSKY, Martin. *The Emotion Machine: Commonsense Thinking, Artificial Intelligence, and the Future of the Human Mind*. 2006. ISBN 0743276639.

[30] CHANDRA, Arijit a Sunil Kumar KHATRI. Spam SMS Filtering using Recurrent Neural Network and Long Short Term Memory. IEEE [online]. 2019 [cit. 10. 4. 2020]. Dostupné z: <https://ieeexplore.ieee.org/abstract/document/9036269>.

[31] GENC, Necmiye. *Detection of spam review on mobile app stores, evaluation of helpfulness of user reviews and extraction of quality aspects using machine learning techniques*. ESPACE [online]. 2019, 225 str. [cit. 10. 4. 2020]. Dostupné z: <https://espace.etsmtl.ca/id/eprint/2463>.

[32] KANAGARAJAH, Archchitha a Eugene Yugarajah Andrew CHARLES. *Opinion Spam Detection in Online Reviews Using Neural Networks*. IEEE [online]. 2019 [cit. 10. 4. 2020]. Dostupné z: <https://ieeexplore.ieee.org/abstract/document/9023695>.

[33] JAIN, Gauri, Manisha SHARMA a Basant AGARWAL. *Spam detection in social media using convolutional and long short term memory neural network*. SPRINGER [online]. 2019 [cit. 10. 4. 2020]. Dostupné z: <https://link.springer.com/article/10.1007/s10472-018-9612-z>.

[34] BARUSHKA, Aliaksandr a Petr HAJEK. *Spam filtering using integrated distribution-based balancing approach and regularized deep neural networks*. SPRINGER [online]. 2018 [cit. 10. 4. 2020]. Dostupné z: <https://link.springer.com/article/10.1007/s10489-018-1161-y>.

[35] WU, Tingmin, Shigang LIU, Jun ZHANG a Yang XIANG. *Twitter spam detection based on deep learning* [online]. 2016 [cit. 10. 4. 2020]. Dostupné z: <https://dl.acm.org/doi/abs/10.1145/3014812.3014815>.

[36] AWAD, Mohammed a Monir FOQAHA. *E-mail spam classification using hybrid approach of RBF Neural Network and Particle Swarm Optimalization: Department of Computer Systems Engineering*. SEMANTICSCHOLAR [online]. 2016 [cit. 10. 4. 2020]. Dostupné z: <https://pdfs.semanticscholar.org/640d/77a1db2f6d03496e40aaa3212e0587ddbd4f.pdf>.

[37] ALKAHT, Issa Joseph a Bassel al KHATIB, 2016. Filtering SPAM Using Several Stages Neural Networks. *International Review on Computers and Software (IRECOS)* [online]. [cit. 21. 12. 2019]. ISSN 18286003. Dostupné z: <https://doi.org/10.15866/irecos.v11i2.8269>.

[38] ZAVVAR, Mohammad, Shole GARAVAND a Meysam REZAEI. *Email Spam Detection Using Combination of Particle Swarm Optimization and Artificial Neural Network and Support Vector Machine*. MECS-PRESS [online]. 2016 [cit. 21. 12. 2019]. ISSN 20750161. Dostupné z: <http://www.mecs-press.org/ijmecs/ijmecs-v8-n7/v8n7-8.html>.

[39] AKSHITA, Tyagi. *Content Based Spam Classification- A Deep Learning Approach* [online]. 2016 [cit. 21. 12. 2019]. Dostupné z: <https://prism.ucalgary.ca/handle/11023/3478>.

[40] WANG, Zhe, William JOSEPHSON, Qin LV a Kai LI. *Filtering Image Spam with Near-Duplicate Detection.* [online]. 2007 [cit. 8. 12. 2019].

Dostupné z: <https://www.researchgate.net/publication/220271864_Filtering_Image_Spam_with_Near-Duplicate_Detection>.

[41] HOWARD, Jeremy a Sebastian RUDER. *Universal Language Model Fine-tuning for Text Classification* [online]. 12 str. [cit. 8. 12. 2019]. Dostupné z: <https://arxiv.org/pdf/1801.06146.pdf>.

[42] CALVO, Miguel Romero. Dissecting BERT Part 1: Understanding the Transformer. *Medium* [online]. [cit. 9. 5. 2020]. Dostupné z: <https://medium.com/@mromerocalvo/dissecting-bert-part1-6dcf5360b07f>.

[43] DEVLIN, Jacob, Ming-Wei CHANG, Kenton LEE a Kristina TOUTANOVA. *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding* [online]. Google AI Language, 16 str. [cit. 9. 5. 2020]. Dostupné z: <https://arxiv.org/abs/1810.04805>.

[44] TRIVEDI, Kaushal. Multi-label Text Classification using BERT — The Mighty Transformer. *Medium* [online]. [cit. 9. 5. 2020]. Dostupné z: <https://medium.com/huggingface/multi-label-text-classification-using-bert/the-mighty-transformer-69714fa3fb3d>.

[45] INGHAM, Francisco. Understanding BERT Part 2: BERT Specifics. *Medium* [online]. [cit. 9. 5. 2020]. Dostupné z: <https://medium.com/dissecting-bert/dissecting-bert-part2-335ff2ed9c73>.

[46] DAR, Pranav. *8 Excellent Pretrained Models to get you Started with Natural Language Processing (NLP)* [online]. 2019 [cit. 21. 12. 2019]. Dostupné z: <https://www.analyticsvidhya.com/blog/2019/03/pretrained-models-get-started-nlp/>.

[47] VASWANI, Ashish, Noam SHAZEER, Niki PARMAR, Jakob USZKOREIT, Llion JONES, Aidan N. GOMEZ, Lukasz KAISER a Illia POLOSUKHIN. *Attention Is All You Need* [online]. [cit. 11. 5. 2020]. Dostupné z: <https://arxiv.org/pdf/1706.03762.pdf>.

[48] Transformers Documentation: BERT. *Huggingface* [online]. [cit. 14. 5. 2020]. Dostupné z: <https://huggingface.co/transformers/index.html>.

[49] RADFORD, Alec, Karthik NARASIMHAN, Tim SALIMANS a Ilya SUTSKEVER. *Improving Language Understanding by Generative Pre-Training* [online]. [cit. 14. 5. 2020]. Dostupné z: <https://cdn.openai.com/research-covers/language-unsupervised/language_understanding_paper.pdf>.

[50] SHUKRI, Mohd. How the Embedding Layers in BERT Were Implemented. *Medium* [online]. [cit. 14. 5. 2020]. Dostupné z: <https://medium.com/why-bert-has-3-embedding-layers/their-implementation-details-9c261108e28a>.

[51] YANG, Zhilin, Zihang DAI, Yiming YANG, Jaime CARBONELL, Ruslan SALAKHUTDINOV a Quoc V. LE. *XLNet: Generalized Autoregressive Pretraining for Language Understanding* [online]. Carnegie Mellon University: Google AI Brain Team, 2020 [cit. 22. 5. 2020]. Dostupné z: <https://arxiv.org/pdf/1906.08237.pdf>.

[52] YANG, Zhilin a Zihang DAI. XLNet: Generalized Autoregressive Pretraining for Language Understanding. *Github* [online]. [cit. 25. 5. 2020]. Dostupné z: <https://github.com/zihangdai/xlnet>.

[53] LIANG, Xu. What is XLNet and why it outperforms BERT: Basic knowledge of XLNet to understand the difference between XLNet and BERT [online]. *Towards data science*, 2019 [cit. 22. 5. 2020]. Dostupné z: <https://towardsdatascience.com/what-is-xlnet-and-why-it-outperforms-bert-8d8fce710335>.

[54] LIANG, Xu. What is Two-Stream Self-Attention in XLNet: Understand the Two-Stream Self-Attention in XLNet [online]. *Towards data science*, 2019 [cit. 22. 5. 2020]. Dostupné z: <https://towardsdatascience.com/what-is-two-stream-self-attention-in-xlnet-ebfe013a0cf3>.

[55] HUANG, Bill. Text Classification with XLNet in Action. *Medium* [online]. [cit. 25. 5. 2020]. Dostupné z: <https://medium.com/yingbiao/text-classification-with-xlnet-in-action-869029246f7e>.

[56] KHAN, Suleiman. BERT, RoBERTa, DistilBERT, XLNet: Which one to use. *Kdnuggets* [online]. [cit. 25. 5. 2020]. Dostupné z: <https://www.kdnuggets.com/2019/09/bert-roberta-distilbert-xlnet-one-use.html>.

[57] WAN, Li, Matthew ZEILER, Sixin ZHANG, Yann LECUN a Rob FERGUS. *Regularization of Neural Networks using DropConnect* [online]. New York [cit. 21. 12. 2019]. Dostupné z: <http://yann.lecun.com/exdb/publis/pdf/wan-icml-13.pdf>.

[58] NABI, Javaid. *Machine Learning: Text Processing* [online]. 2018 [cit. 18. 12. 2019]. Dostupné z: <https://towardsdatascience.com/machine-learning-text-processing-1d5a2d638958>.

[59] THOMAS, Rachel. *Language Modeling and Sentiment Analysis of IMDB movie reviews* [online]. [cit. 19. 12. 2019]. Dostupné z: <https://github.com/fastai/course-nlp/blob/master/5-nn-imdb.ipynb>.

[60] OLAH, Christopher. *Understanding LSTM Networks* [online]. Canada, 2015 [cit. 21. 12. 2019]. Dostupné z: <https://colah.github.io/posts/2015-08-Understanding-LSTMs/>.

[61] MERITY, Stephen. *The WikiText Long Term Dependency Language Modeling Dataset: Wikitext 103* [online]. [cit. 18. 12. 2019]. Dostupné z: <https://blog.einstein.ai/the-wikitext-long-term-dependency-language-modeling-dataset/>.

[62] NABI, Javaid. *Machine Learning: Text Classification, Language Modelling using fast.ai* [online]. 2019 [cit. 18. 12. 2019]. Dostupné z: <https://towardsdatascience.com/machine-learning-text-classification/language-modelling-using-fast-ai>.

[63] HOWARD, Jeremy a Rachel THOMAS. *Fastai for PyTorch* [online]. 2019 [cit. 18. 12. 2019]. Dostupné z: <https://www.fast.ai/>.

[64] GHELANI, Shreya. *From Word Embeddings to Pretrained Language Models: A New Age in NLP (Part 1)* [online]. 2019 [cit. 18. 12. 2019]. Dostupné z: <https://towardsdatascience.com/from-word-embeddings-to-pretrained-language-models/a-new-age-in-nlp-part-1-7ed0c7f3dfc5>.

[65] GHELANI, Shreya. *From Word Embeddings to Pretrained Language Models: A New Age in NLP (Part 2)* [online]. 2019 [cit. 20. 12. 2019]. Dostupné z: <https://towardsdatascience.com/from-word-embeddings-to-pretrained-language-models/a-new-age-in-nlp-part-2-e9af9a0bdcd9>.

[66] SMITH, Leslie N. Disciplined approach to neural network hyper-parameters: Part 1 — learning rate, batch size, momentum, and weight decay. *US Naval Research Laboratory* [online]. Washington, USA, 2018 [cit. 18. 12. 2019]. Dostupné z: <https://arxiv.org/pdf/1803.09820.pdf>.

[67] SHAH, Tarang. *About Train, Validation and Test Sets in Machine Learning* [online]. [cit. 19. 12. 2019]. Dostupné z: <https://towardsdatascience.com/train-validation-and-test-sets-72cb40cba9e7>.

Seznam symbolů, veličin a zkratek

AELM Autoencoder Language Model – Druh jazykového modelu
AI Artificial Intelligence – Umělá inteligence
ANN Artificial Neural Network – Umělá neuronová síť
API Application Programming Interface – Rozhraní pro programování
ARPANET Advanced Research Projects Agency Network – První počítačová síť
ARLM Autoregressive Language Model – Autoregrisivní jazykový model
ASCII American Standard Code for Information Interchange – Znaková sada
BANN Bidirectional Artificial Neural Network – Obousměrná neuronová síť
BERT Bidirectional Encoder Representations from Transformers
Bi-LSTM Bi-directional Long-Short Term Memory – Druh neuronových sítí
CNN Convolutional Neural Networks – Konvoluční neuronové sítě
CPU Central Processing Unit – Centrální procesorová jednotka
CRF Conditional Random Field – Statistická metoda modelování
CSA Content Stream Attention – Technika používána v transformer sítích
CSS Cascading Style Sheets – Kaskádové styly
DBB Distribution-Based Balancing – Optimalizační algoritmus
DBM Deep Boltzmann Machine – Hluboký Boltzmannův stroj
DBN Deep Belief Network – Hluboká síť důvěry
DBT Deep Bidirectional Transformers – Hluboké obousměrné transformery
DCNN Deep Convolutional Neural Network – Hluboká konvoluční síť
DMLP Dense Multi Layer Perceptron – Druh hlubokých neuronových sítí
DNN Deep Neural Network – Hluboké neuronové sítě
DNS Domain Name System – Systém doménových jmen
EMAIL Electronic mail – Elektronická pošta
GLMP General Language Model Pre-trained – Jazykový model
GloVe Global Vectors for Word Representation – Technika vnoření slov
GPU Graphics Processing Unit – Grafická výpočetní jednotka
HAM Solicited electronic messages – Vyžadovaná pošta
HTLM HyperText Markup Language – Hypertextový značkovací jazyk
HTTP Hypertext Transfer Protocol – Internetový protokol
IMAP Internet Message Access Protocol – Protokol na stažení e-mailů
LMT Logistic Model Tree – Druh rozhodovacího stromu
LSTM Long Short-Term Memory – Druh architektury RNN
MIME Multipurpose Internet Mail Extensions – E-mailové rozšíření
ML Machine Learning – Strojové učení
MLM Masked Language Modeling – Maskované jazykové modelování
MLP Multi Layer Perceptron – Druh neuronové sítě

NBTree	Naïve Bayes Tree – Naivní Bayes stromy
NLP	Natural language processing – Zpracování přirozeného jazyka
NN	Neural Network – Neuronová síť
NSP	Next Sentence Prediction – Technika predikování další věty
PLM	Permutation Language Modelling – Permutační funkce modelování
POP3	Post Office Protocol – Protokol na stažení elektronické pošty
PSO	Particle Swarm Optimalization – Technika extrahování příznaků
QA	Question Answering – Zodpovězení na otázky, úloha NLP
QSA	Query Stream Attention – Technika požívána v síti `XLNet`
RB	Radial Basis – Druh aktivační funkce
RBFNN	Radial Basis Function Neural Networks – Druh neuronových sítí
RF	Random forests – Náhodné lesy
RNN	Recurrent Neural Network – Rekurentní neuronové sítě
SMTP	Simple Mail Transfer Protocol – Protokol na předání e-mailů
SNN	Shallow Neural Network – Mělké neuronové sítě
SPAM	Unsolicited electronic messages – Nevyžádaná pošta
SVM	Support Vector Machine – Metoda podpůrných vektorů
TSSA	Two-Stream Self-Attention – Druh mechanismu pozornosti
TWE	Transferring Word Embeddings – Technika vnoření
ULMFiT	Universal Language Model Fine-tuning – Technika strojového učení
XLNet	Generalized Autoregressive Pretraining – Druh neuronové sítě

Seznam příloh

A Návod na spouštění parsovacích skriptů

První praktický výstup této diplomové práce je zastoupen Python programem, který umožňuje provádět parsování surových e-mailů ve formátech EML a DAT. Po provedení všech kroků zpracování e-mailu zahrnující jeho dekódování, odstranění poškozených anebo nepodstatných pro analýzu částí, ověřování jazyku apod., je zpracovaná elektronická zpráva uložena do finálního CSV dokumentu a následně je připravena na napojení na vstup do vybraných druhů neuronových sítí. Napojení CSV dokumentu a jeho finální přizpůsobení do podoby vhodné k modelování, tokenizaci a numerikalizaci je docíleno skripty, které jsou uvedené v příloze B. Spouštění parsovacích přípravných skriptů je možné buď z přiloženého optického media (viz adresářová struktura přiloženého optického media), anebo přímo z otevřeného repositáře GitHub[1]. V případě, kdy je spouštění kódu plánováno provádět přes webové rozhraní GitHub, je nutné nainstalovat nástroj git[2] a přes příkazovou řádku provést stažení složky projektu této diplomové práce:

```
git clone https://github.com/ysafonov/AIEmailSpamFiltering.git
```

V dalším kroku je nutné nainstalovat samotný Python[3], který byl základním programovacím jazykem pro vytvoření praktické části této práce. Za zmínku stoji skutečnost, že při vytvoření a otestování kódu byla použitá poslední stabilní verze Python, konkrétně se jedná o verzi 3.8.0. Zjistit nainstalovanou verzi Python lze pomocí konzolového příkazu:

```
python --version
```

V dalším kroku je nutné pomocí nástroje pip[4] nainstalovat externí knihovny, které jsou použity pro zpracování elektronických zpráv. Docílit nainstalování je možné pomocí sady příkazů:

```
pip install langdetect
pip install html2text
pip install chardet
pip install numpy
pip install pandas
```

V daném kroku je dobré zmínit, že vývoj a testování kódu bylo uskutečněné ve virtuálním prostředí Conda, jehož primárním účelem je oddělení různých Python balíčků

[1] Jedná se o otevřený GitHub repositář, který je dostupný z URL: <https://github.com/ysafonov/AIEmailSpamFiltering.git>.

[2] Nástroj git je pro operační systém Windows dostupný z oficiálních stránek společnosti, viz URL: <https://git-scm.com/download/win>.

[3] Python je dostupný z oficiálních stránek: <https://www.python.org/downloads/>.

[4] Kompletní návod na instalaci nástroje pip lze zjistit z oficiálních stránek: <https://pip.pypa.io/en/stable/installing/>.

a knihoven v něm nainstalovaných. Použití `Conda`[5] umožňuje docílit transparentnosti v procesu aktualizace balíčků projektu a pomáhá se vyhnout různým konfliktům, které mohou byt spojené s verzováním a jejich kompatibilitou. Za zmínku stojí, že použití a instalace virtuálního prostředí `Conda` je doporučeno nikoliv vyžadováno. Úspěšné spouštění kódu parseru lze docílit i bez splnění výše popsáného kroku.

Po instalaci všech výše zmíněných nástrojů a knihoven je skript vytvořený v rámci diplomové práce připraven k použití. Princip fungování vytvořeného scriptu lze zjistit z vytvořené nápovědy, kterou lze zobrazit následujícím příkazem[6]:

```
MainConsoleScriptEmailParser.py --help
```

Po spuštění souboru se uživateli programu zobrazí nápověda, viz obrázek číslo A.1. Důležité je zmínit, že vytvořený skript umí pracovat ve dvou různých módech.

```
D:\AIEmailSpamFiltering\source\preparation> MainConsoleScriptEmailParser.py --help

Information about parser usage:
[--help or -h information about scipt usage]
[--singlemode or -s run the script in the single mode]
[--directory <path> or -d <path> emils data directory]
[--mail <name> or -m <name> contains email's name with extensions (only single mode)]
[--csv <name> or -c <name> output file name]
[--language <short name> language emails filtering: en,cz,ru,sk]
[-v <number> the maximal number of email's entities inside an output csv]
```

Obr. A.1: Výpis nápovědy při použití vytvořeného parseru

První mód je aktivován pomocí argumentu `-singlemode` anebo `-s` a slouží k parsování ojedinělých e-mailů. Pro úspěšné fungování daného módu je důležité zadat správnou cestu k umístěnému e-mailu, včetně jeho přípony. Druhý mód je aktivní defaultně a slouží pro dávkové zpracování všech e-mailů uložených v cílové složce. Pro jeho úspěšné fungování je nutné e-mailové entity umístit do existujícího podadresáře `Data` v projektu, viz strukturu projektu, anebo definovat vlastní složku se zachováním stejné adresářové struktury. Po ukončení své činnosti skript roztřídí vstupní e-maily do složek (`Parsed`, `Exception` a `LanException`), vytvoří finální CSV dokument požadované velikosti (argument `-v`) a obsahující e-maily ve zvolených jazycích (argument `-language`). Detailnější informace o principech fungování skriptu je možné zjistit z dokumentace vytvořené při napsání zdrojového kódu.

[5]Kompletní návod na instalaci nástroje `Conda` a jeho použití lze zjistit z oficiálních stránek: <`https://docs.conda.io/en/latest/`>.

[6]Příkaz je nutné vykonávat přímo ze složky se zdrojovými kódy projektu, konkrétně se jedná o adresář: `/source/preparation/`. Navíc jako argument pro zobrazení nápovědy lze použit argument `-help`, anebo jeho zkrácenou alternativu v podobě argumentu `-h`, viz obrázek A.1.

B Návod na zprovoznění neuronových sítí

Proces tvorby neuronových sítí spočíval v jejich vývoji pomocí `Jupyter Notebook`. Použití dané technologie umožňuje efektivněji přistoupit k trénování neuronových sítí a přehledněji zobrazit dosažené výsledky. Základní testování se provádělo ve volně dostupném prostředí `Google Colaboratory`[1] a následně, po úspěšném dokončení dané fáze, byly modely přeneseny na reálný hardware, kde fáze učení byla dokončena. Konkrétně se jednalo o grafickou kartu `Nvidia Xp` s podporou architektury CUDA a kapacitou paměti 12 196 MiB typu GDDR5X.

Modely neuronových sítí je tedy možné zprovoznit třemi různými způsoby, které byly seřazeny vzestupně na základě jejich obtížnosti jimi kladených podmínek.

1. Spouštění kódu ve virtuálním prostředí `Google Colaboratory`;
2. Zprovoznění modelů na lokálním hardware pomocí `Jupyter Notebook`;
3. Zprovoznění modelů na lokálním hardware pomocí `Python` skriptu.

První způsob otestování modelů lez považovat za nejjednodušší. Princip spočívá v nahrání do virtuálního prostředí `Google Colaboratory` zapsaných na optické medium `Jupyter` sešitů (viz složka `/notebook/`). Pro zjednodušení lze v `Google Colaboratory` importovat kód projektu přímo z verzovacího systému `GitHub`. `Google Colaboratory` automaticky odhalí soubory s rozšířením `ipynb` a požádá uživatele o zvolení konkrétního souboru na spuštění. Pro lepší orientaci ve vytvořeném kódu byly etapy trénování jednotlivých modelů neuronových sítí rozděleny do příslušných složek a detailně okomentovány, viz adresářová struktura optického média. Nevýhoda daného způsobu spočívá v nutnosti nahrávat datové rámce nevyžádané a legitimní pošty zvlášť a upravovat cesty ukládání a nahrávání příslušných souborů. Za zmínku stoji skutečnost, že po prvotním testování kódu přeneseného učení neuronových sítí byl kód plně přenesen a vyvíjen na reálném Linux serveru, a proto plná funkčnost modelů v `Google Colaboratory` není garantována. Dalším omezením tohoto způsobu je nedostatek kapacit virtuálního GPU na provedení plnohodnotného trénování. V rámci testováni kódu často docházelo k chybám nedostatků paměti, přičemž nebyly vneseny žádné změny do kódu projektu.

Druhý a garantovaně funkční způsob otestování modelu je možný pomocí `Jupyter Notebook`. Daný způsob vyžaduje od uživatele disponovat výkonnou grafickou kartou[2] značky `Nvidia`, která je postavená na architektuře CUDA a mít nainstalované knihovny a ovladače pro provedení procesu učení neuronových sítí. V případě odlišnosti parametrů paměti dostupného hardware je po instalaci všech knihoven možné

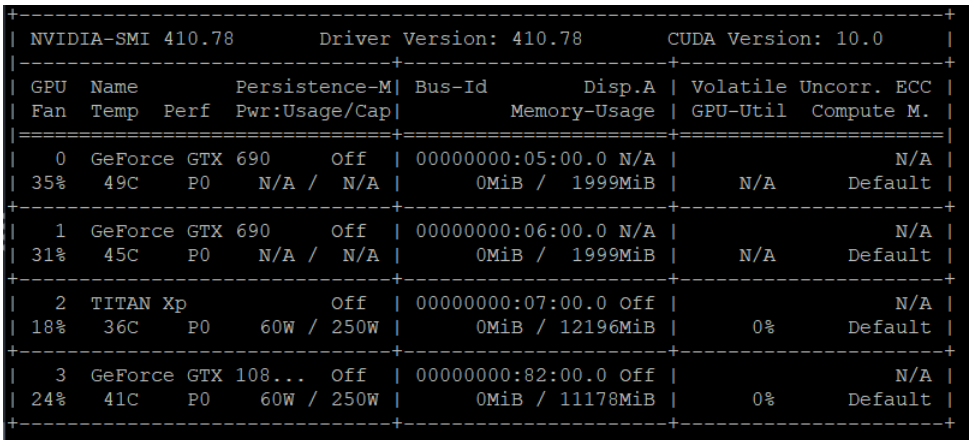
[1] Virtuální prostředí na vývoj aplikací je vyvinuto společností Google a je dostupné z URL: <`https://colab.research.google.com/`>.

[2] Podle názoru autora této diplomové práce je pro úspěšně trénování modelů neuronových sítí nutné zajistit minimálně takový hardware, který byl použit při provedení experimentu.

přizpůsobit parametry učení sítí (viz podkapitola číslo 3.3) v podobě `batch_size`, `MAX_LEN` apod. Sledovat vytížení grafické karty je možné pomocí GPU management nástroje `nvidia-smi`[3], a to pomocí příkazu:

```
nvidia-smi
```

Konzolový výpis daného nástroje je pro server použitý v experimentu `Gravy` zobrazen na obrázku číslo B.1.



Obr. B.1: Konzolový výpis příkazu `nvidia-smi`

V dalším kroku bude zapotřebí nainstalovat prostředí `Jupyter Notebook` a speciální nástroj `ipywidgets` sloužící pro stažení externích souboru přímo ze spuštěného sešitu. Provést zmíněné kroky je možné pomocí příkazů:

```
pip install notebook
pip install ipywidgets
```

Aktivace příslušného `Jupyter` serveru se provádí přímo ze složky projektu (konkrétně se jedná o složku `AIEmailSpamFiltering`) pomocí příkazu:

```
jupyter notebook
```

Defaultně `Jupyter` server naslouchá na portu 8888 a pro komunikaci používá nezabezpečený protokol HTTP[4].

V dalším kroku je nutné nainstalovat knihovny pro spouštění neuronových sítí. Je podstatné zmínit, že nutnost doinstalování `Python` knihoven se může lišit v závislosti

[3]Kompletní návod GPU management nástroje `nvidia-smi` a jeho použití lze zjistit z oficiálních stránek společnosti: <`https://developer.nvidia.com`>.

[4]V rámci testování modelů, konfigurační soubory `Jupyter` serveru byly změněny, a to takovým způsobem, aby komunikace a vzdálené spouštění kódu probíhalo maximálně bezpečně. Server v rámci experimentu komunikoval pomocí protokolu HTTPS (byl vytvořen *self-signed* certifikát) na portu 8889 a pro otevření příslušných `Jupyter` sešitů byla vyžadována autentizace uživatele.

na používaných ovladačích a už nainstalovaných balíčcích. Minimálně nutný seznam balíčku lze stáhnout pomocí nástroje `pip` příkazy[5]:

```
pip install pytorch==1.2.0
pip install torchvision==0.4.0
pip install cudatoolkit==10.0
```

V další etapě dojde bezprostředně k instalaci základních modelů neuronových sítí. Pro oddělení nainstalovaných balíků pro modely `ULMFiT`, `BERT` a `XLNet` lze použit virtuální prostředí `Conda`, viz kapitola A.

Pro zprovoznění modelu neuronové sítě `ULMFiT` je nutné nainstalovat knihovnu `fast.ai`[6] pomocí příkazu:

```
pip install fastai
```

Pro model `BERT` je zapotřebí stáhnout dvě knihovny pomocí příkazů:

```
pip install pytorch-pretrained-bert
pip install pytorch-nlp
```

Pro model `XLNet` byla použita oficiální knihovna `transofrems`[7], jejíž instalaci lze provést příkazem:

```
pip install transformers
```

Po úspěšném nainstalování všech potřebných balíků, konfiguraci `Jupyter` serveru, budou naprogramované `Jupyter` sešity plně funkční. Za zmínku stojí skutečnost, že zdrojový kód `Jupyter` sešitů byl vytvořen na základě oficiálních dokumentů a návodů, jejichž kompletní seznam lze zjistit ze zdrojového kódu modelů a také z literatury této diplomové práce. Navíc k poskytovanému kódu byly pečlivě sepsány komentáře, a to s motivací lepšího pochopení neodborníky a efektivnější údržby kódu z dlouhodobého hlediska.

Třetí způsob spouštění modelů lze provést přímo z příkazové řádky bez přímého spouštění `Jupyter` serveru, a to pomocí tříd automaticky vygenerovaných programem Visual Studio Code v jazyce `Python`, viz složka `/source/learning/`.

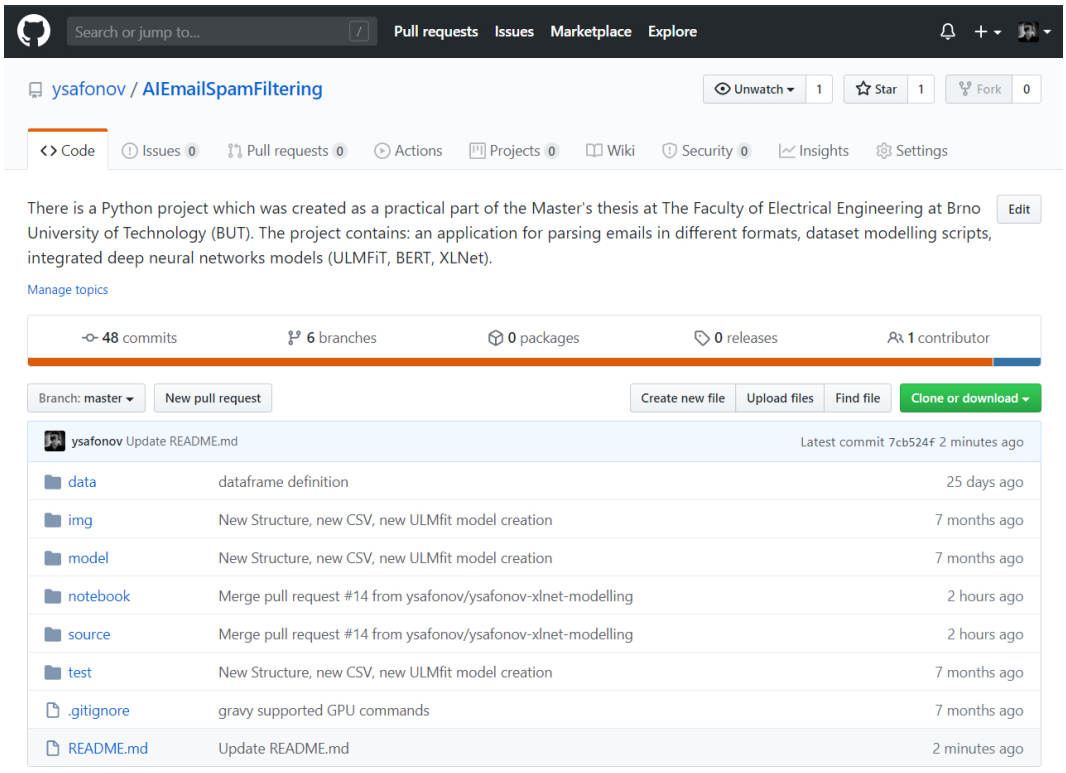
[5]Zmíněné verze nainstalovaných knihoven byly použity při spouštění kódu na cílovém serveru `Gravy` a mohou se v závislostí na nainstalovaných CUDA ovladačích lišit.

[6]Knihovna `fast.ai` a její dokumentace je dostupná z URL: <`https://www.fast.ai/`>.

[7]Dokumentace ke knihovně `transofrems` je dostupná z URL: <`https://huggingface.co/transformers/model_doc/xlnet.html`>.

C Obsah přiloženého media

K textu diplomové práce se přikládá optické medium, které obsahuje všechny materiály důležité pro úspěšné spuštění e-mailového parseru a ověření funkčnosti v textu popsaných neuronových sítí. Z poskytnutého zdrojového kódu je možné provést vlastní epochy trénování modelů, anebo pro vyřešení úlohy klasifikace e-mailového spamu použit již uložené modely `ULMFiT`, `BERT` a `XLNet`. Přiložené DVD medium obsahuje kromě zdrojového kódu vytvořeného parseru a neuronových sítí také elektronickou verzi diplomové práce a zpracovanou datovou sadu legitimních e-mailů. Za zmínku stojí, že datová sada nevyžádaných e-mailů byla zašifrována, a to z důvodu bezpečnostní politiky společnosti TrustPort a podmínek poskytnutí elektronických zpráv. Klíč pro dešifrování datové sady byl předán vedení společnosti TrustPort. Navíc byl vývoj zdrojového kódu uskutečněn jak již bylo zmíněno výše pomocí verzovacího systému `GitHub`. Ve veřejném repositáři[1] je možné najít komentáře, návod ke spuštění a historii změn všech částí kódu poskytnutého na DVD mediu. Strukturu přiloženého DVD media se stručnými komentáři lze vidět na další stránce.



Obr. C.1: Stránka zdrojového kódu diplomové práce na `GitHub`

[1] Jedná se o otevřený `GitHub` repositář, který je dostupný z URL: <`https://github.com/ysafonov/AIEmailSpamFiltering.git`>.

Adresářová struktura přiloženého optického media je tedy následující:

- `/` — kořenový adresář optického DVD media
  - `AIEmailSpamFiltering/` — zdrojový kód práce
    - `img/` — grafické podklady projektu
    - `data/` — datová sada použitá při učení modelů
      - `processed/` — složka obsahující zpracované e-maily v CSV formátu
        - `ham` — složka zastupující ham CSV dokumenty
        - `spam` — složka zastupující spam CSV dokumenty
      - `raw/` — složka obsahující surové e-mailové entity
        - `ham` — surové ham e-maily
        - `spam` — surové spam e-maily
    - `model/` — adresář s uloženými modely neuronových sítí
      - `ulmfit/` — adresář s modely `ULMFiT`
      - `bert/` — adresář s modely `BERT`
      - `xlnet/` — adresář s modely `XLNet`
    - `notebook/` — složka s naprogramovanými `Jupyter` sešity
      - `ulmfit/` — sešity modelu `ULMFiT`
        - `ulmfit-01-language-model-creation.ipynb` — první fáze
        - `ulmfit-02-learner-language-model-first-cycle.ipynb` — druhá fáze
        - `ulmfit-03-learner-language-model-tuning.ipynb` — třetí fáze
        - `ulmfit-04-tuned-language-model-testing.ipynb` — čtvrtá fáze
        - `ulmfit-05-spam-classifier-tuning.ipynb` — pátá fáze
        - `ulmfit-06-spam-classifier-testing.ipynb` — šestá fáze
      - `bert/` — sešit modelu `BERT`
        - `bert-datamodeling-learning-testing.ipynb` — fáze trénování
      - `xlnet/` — sešit modelu `XLNet`
        - `xlnet-datamodeling-learning-testing.ipynb` — fáze trénování
    - `source/` — adresář se zdrojovými kódy
      - `learning/` — automaticky vygenerované kódy na základě sešitů
      - `modeling/` — finální transformace a znáhodnění dat
        - `MainScriptDataModeling.py` — hlavní skript fáze transformace
        - `ScriptDataModelingRandomization.py` — třída zpracování CSV entit
      - `preparation/` — adresář s kódem parseru
        - `EmailDataSetProcessing.py` — ukládaní CSV a statistik po zpracování
        - `EmailParsing.py` — zpracování předmětu a těla e-mailu
        - `StaticLanguageClassification.py` — vyhodnocení jazyku
        - `MainConsoleScriptEmailParser.py` — hlavní skript parseru
      - `processing/` — třídy sloužící k přizpůsobení dat pro modelování
        - `UlmfitDataProcessing.py` — příprava dat pro `ULMFiT`
        - `BertDataProcessing.py` — příprava dat pro `BERT`
        - `XLNetDataProcessing.py` — příprava dat pro `XLNet`
        - `MainScriptUlmfitDataProcessing.py` — skript zpracování
        - `MainScriptBertDataProcessing.py` — skript zpracování
        - `MainScriptXLNetDataProcessing.py` — skript zpracování
  - `YehorSafonovMasterThesis.pdf` — textový podklad diplomové práce