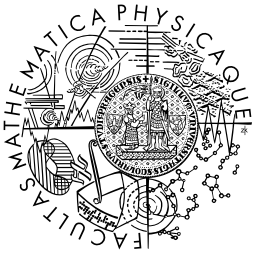


MATEMATICKO-FYZIKÁLNÍ FAKULTA
Univerzita Karlova

DIPLOMOVÁ PRÁCE

Bc. Patrik Romanský

Automatická detekcia fake-news na slovenských textoch

Ústav formální a aplikované lingvistiky

Vedoucí diplomové práce: RNDr. David Mareček, Ph.D.
Studijní program: Informatika
Studijní obor: Umělá inteligence

Praha 2023

Prohlašuji, že jsem tuto diplomovou práci vypracoval(a) samostatně a výhradně s použitím citovaných pramenů, literatury a dalších odborných zdrojů. Tato práce nebyla využita k získání jiného nebo stejného titulu.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona v platném znění, zejména skutečnost, že Univerzita Karlova má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle §60 odst. 1 autorského zákona.

V ..... dne ....

Podpis autora

Chcel by som vyjadriť veľkú vďaku všetkým, ktorí mi pomohli pri tvorbe tejto diplomovej práce. V prvom rade by som sa chcel poďakovať môjmu vedúcemu práce RNDr. David Mareček, Ph.D. za jeho odborné vedenie, rady a podporu počas celej práce. Ďalej by som sa chcel poďakovať Technickej univerzite v Košiciach a tímu Sarnovkého za ich ochotu poskytnúť mi dátovú sadu a tým umožniť experimentovanie na slovenskom datasete.

Taktiež sa chcem veľmi poďakovať mojej rodine a všetkým milovaným, ktorí mi poskytli obrovskú podporu počas celého môjho života a podnietili ma k štúdiu na Matematicko-Fyzikálnej Fakulte Univerzity Karlovej, čo viedlo k napísaniu tejto diplomovej práce. Rovnako by som chcel vyjadriť obrovskú vďaku za ich nepretržitú oporu a pomoc pri písaní a korektúre tejto práce. Osobitná vďaka patrí mojím najbližším rodičom a sestre, bez ktorých by som to nezvládol. Veľmi pekne ďakujem, že pri mne stojíte.

Nakoniec, no rozhodne nie s menšou dôležitosťou, by som chcel vyjadriť svoju vďaku všetkým mojim priateľom, ktorí boli so mnou počas mojich rokov na univerzite. Špecialne by som sa chcel poďakovať môjmu najlepšiemu priateľovi a spolužiakovi Matúšovi, ktorého som spoznal na vysokej škole. Boli sme si navzájom oporou pri štúdiu a spoločne sme prežili najkrajšie chvíle našej mladosti. Tiež mu patrí veľká vďaka za pomoc s korektúrou tejto diplmovej práce.

Ešte raz by som chcel vyjadriť veľkú vďaku všetkým, ktorí mi podali pomocnú ruku pri štúdiu a písaní tejto diplomovej práce. Ďakujem.

Název práce: Automatická detekcia fake-news na slovenských textoch

Autor: Bc. Patrik Romanský

Katedra: Ústav formální a aplikované lingvistiky

Vedoucí diplomové práce: RNDr. David Mareček, Ph.D., Ústav formální a aplikované lingvistiky

Abstrakt: Šírenie fake-news je dlhodobým problémom, ale v posledných rokoch sa stáve ešte výraznejším. Preto sme sa v tejto práci pozreli na problém ich automatickej detekcie ako na úlohu klasifikácie textu. Práca sa od iných, jej podobných štúdií, odlišuje primárne v tom, že sa zameriava na slovenčinu, kde doposiaľ nebola vykonaná takáto rozsiahla sada experimentov. Počas testov sme vytvorili vybalansovaný dataset. Vykonali sme taktiež viac ako 80 experimentov s cieľom nájsť optimálny klasifikátor pre riešenie tohto problému. Ako prvý sme použili predtrénované jazykové modely typu Transformer (BERT, mBERT, RoBERTA, XLM-RoBERTa a SlovakBERT) a pomocou štandardných metrík sme porovnali ich výkonnosť s inými metódami strojového učenia. Pre fine-tuning sme použili aj anglické datasety LIAR a COVID19 FN, na ktorých sme otestovali vplyv témy fake-news a prenos vlastnosti medzi jazykmi. Najlepšie výsledky dosiahol SlovakBERT v kombinácii s tréningom na výlučne slovenskom datasete ($acc = 0.9610$).

Klíčová slova: falošné správy, hoax, NLP, SlovakBERT,

Title: Automatic detection of fake-news on Slovak texts

Author: Bc. Patrik Romanský

Department: Institute of Formal and Applied Linguistics

Supervisor: RNDr. David Mareček, Ph.D., Institute of Formal and Applied Linguistics

Abstract: Fake news is a problem in recent years. This study focuses on detecting fake news written in the Slovak language using text classification methods. It is unique because it is the first to conduct such a comprehensive set of experiments on Slovak. During the study, a balanced dataset was created, and over 80 experiments were conducted to find the optimal classifier for the problem. Pre-trained transformer-based language models, including BERT, mBERT, RoBERTA, XLM-RoBERTa, and SlovakBERT, were used in the initial step of the study, and their performance was compared against other machine learning methods using standard metrics. The models were fine-tuned with LIAR and COVID19 FN, English-language datasets, to test the impact of fake news topics and language transfer properties. SlovakBERT combined with training exclusively on Slovak datasets achieved the best results with an ($acc = 0.9610$). This study can contribute to the development of tools to automatically detect fake news in Slovak, aiding in the fight against the spread of false information.

Keywords: fake-news, hoax, NLP, SlovakBERT

Obsah

Úvod

V súčasnosti je veľké množstvo informácií dostupných prostredníctvom tradičných médií a sociálnych sietí. Výhody digitálnej éry a vysokorýchlostného internetu prinášajú množstvo príležitostí, ale zároveň aj riziká. Jedným z najväčších nebezpečenstiev, ktoré môžu vyplynúť z nesprávneho a zavádzajúceho šírenia informácií, sú falošné správy (angl. fake-news). Tieto správy môžu mať vplyv na verejnú mienku a dokonca ovplyvniť výsledky politických volieb a spôsobiť značné škody na povesti jednotlivcov alebo inštitúcií.

S narastajúcim množstvom falošných správ sa stáva čoraz dôležitejšie mať k dispozícii nástroje, ktoré dokážu automaticky detekovať tieto nepravdivé informácie. V súčasnosti existuje množstvo technológií a algoritmov, ktoré sa používajú na detekciu fake-news v rôznych jazykoch. Avšak väčšina týchto riešení sú zamerané na anglický jazyk a pre slovenské jazykové prostredie nie sú úplne vhodné. Preto je potrebné vytvoriť model, ktorý dokáže efektívne detekovať falošné správy v slovenských textoch.

Vzhľadom na vyššie uvedené faktory sa stalo naliehavou úlohou vytvoriť automatické nástroje pre slovenský jazykový korpus, ktoré dokážu odhaliť a klasifikovať takéto falošné správy, pričom je dôležité venovať pozornosť času od zverejnenia falošných správ po ich odhalenie a zmazanie. Veľmi často je dôležité vymazať takéto správy čo najskôr, aby sa zabránilo ich šíreniu. V takýchto časovo kritických prípadoch nemusí byť dostatok času na manuálne overenie pravdivosti informácii vo veľkom množstve článkov, ktoré sú v dnešnej dobe zdieľané.

V digitálnom veku je šírenie dezinformácií lacnejšie a jednoduchšie. V minulosti iba veľké médiá s dostatočnými zdrojmi boli schopné osloviť veľké publikum. Dnes, môže byť každý producentom pre široké masy [1]. Preto sa v tejto práci zameriame na tento moderný fenomén šírenia falošných správ a ich automatickej detekcie pomocou metód strojového učenia.

Cieľ práce

Táto práca sa zaoberá úlohou odhaľovania falošných správ ohľadom covid-19 ako problémom klasifikácie textu s cieľom zistiť, či je možné falošné správy odhaliť výlučne na základe špecifík použitého jazyka. Zameriava sa na algoritmy strojového učenia a ich porovanie úspešnosti na testovaných slovenských datasetoch.

1. Vytvorenie alebo získanie slovenského datasetu, ktorý v sebe bude obsahovať dáta covid-19 Fake-news
2. Klasifikácia covid-19 datasetu s aplikovaním poznatkov z iných jazykov
3. Porovnanie rôznych klasifikačných nástrojov vzhľadom k ich úspešnosti a časovej náročnosti tréningu

1. Súvisiace práce

1.1 COVID-19 Fake-News datasety

Pre túto prácu najrelevantnejším datasetom, ktorý je dostupný na internete je Slovak-FakeNews dataset, ktorý bol vytvorený Strakovským a kolektív [2]. Tento dataset je jedným z mála dostatočne roziahlých datasetov pre detekciu falošných správ v slovenskom jazyku. Dátova sada pozostáva z 13 736 článkov, z ktorých je 851 nepravdivých. Dataset bol anotovaný automaticky a následne manuálne onanotovaný 5 hodnotiteľmi. Obsahuje iba textové údaje. Téma falošných správ, ktoré tvoria tento dataset je ohľadom pandámie covid19, preto aj zahraničné datasety boli vyhľadávane z tejto oblasti.

Podľa informácii, ktoré sme získali existujú štyri vhodné a verejne dostupné datasety údajov na detekciu falošných správ v angličtine, z toho tri v oblasti covid-19. Datasety sa od seba odlišujú typom zdroja z akého pochádzajú (sociálne siete, novinové články), dĺžkou textových záznamov, nerovnováhou rozdelenia dát do tried a typom informácie dostupnej pre klasifikáciu (jazykovy/sociálny kontext alebo oba typy informácie). Dataset s názvom Covid19 FN, ktorý bol vyvinutý Patwom a jeho tímom [3], predstavuje prvý zahraničný zdroj na detekciu falošných správ o Covid-19. Obsahuje 5 600 overených tweetov z dôveryhodných zdrojov, ako sú vládne účty, lekárske inštitúcie a spravodajské kanály, ako aj 5 100 manuálne overených nepravdivých tvrdení z rôznych zdrojov, vrátane novinových článkov, tlačových správ a príspevkov na sociálnych sieťach.

FakeCovid je ďalším datasetom určeným pre úlohu klasifikácie COVID-19 fake-news [4]. Narozdiel od predchádzajúceho datasetu je viacjazyčný. Pozostáva z 5 182 manuálne kontrolovaných spravodajských článkov zo 40 jazykov, z ktorých 2 116 je angličtina a 47 nemčina.

Ďalším datasetom pre angličtinu je CoAID [5] obsahuje rôzne typy správ, t. j. tvrdenia, spravodajské články a príspevky na sociálnych sieťach. Konkrétne pozostáva z 255 falošných a 3996 skutočných a overených spravodajských článkov týkajúcich sa pandémie, 28 nesprávnych a 454 pravdivých tvrdení o víruse, 926 príspevkov na sociálnych sieťach, ktoré reagujú na tieto články, Tweety obsahujú, na ktoré tieto správy reagujú, ako aj údaje o používateľovi, plus interakcie s každým tweetom. Keďže až 94 % dát predstavuje autentické správy, je tiež najnevyváženejším z datasetov. Označenia pravdivosti pre skutočné spravodajské články boli získané na úrovni vydavateľov, t. j. údaje boli zozbierané od spoľahlivých vydavateľov, ako sú ScienceDaily alebo WHO. Falošné články boli zhromaždené pomocou odkazov na nepravdivé články z webových stránok overujúcich fakty. V prípade tvrdení sa na identifikáciu vyhlásení, o ktorých je známe, že sú nepravdivé, použili informácie WHO a ďalšie spoľahlivé zdroje [5].

Posledným z anglických datasetov, ktorý chceme predstaviť je LIAR, Dátová sada bola vytvorená výskumníkmi Univerzity v Kalifornii v Berkeley a má slúžiť na tréning a hodnotenie modelov, ktoré dokážu rozpoznať falošné správy [6]. Tento dataset sa od ostatných odlišuje témou článkov, ktorou sú voľby v USA.

Nielenže sú existujúce súbory údajov vzácne, ale sú tiež obmedzené, pokiaľ ide o množstvo textu a typy informácií, ktoré poskytujú, a niekedy sú veľmi nevyvážené, čo spôsobuje skreslenie.

Datasety				
Názov	Jazyk	Clekovo	Pravdivé	Falošné
Covid19 FN	anglický	10700	5600	5100
FakeCovid	viacjazyčný	5182	4146	1036
CoAID	anglický	4251	3996	255
LIAR	anglický	12836	8318	4518
Slovak-FakeNews	slovenský	13736	12885	851

Tabuľka 1.1: Porovnanie datasetov.

1.2 Detekcia COVID-19 Fake-News

Vzhľadom na povahu dostupných datasetov o COVID-19 bola väčšina pokusov o odhalenie falošných správ v tejto doméne založená na jazyku. Prístupom s najlepšími výsledkami v zdieľanej úlohe COSTRAINT 2021 Workshop on Combating Online Hostile Posts in Regional Language during Emergency Situation, na datasete o falošných správach Covid19 navrhli Glazková a kolektív [7], ktorí získali textové reprezentácie pomocou jazykového modelu COVID-Twitter-BERT [8], ktorý bol špeciálne vyškolený na 160 miliónoch anglických tweetov o víruse a dosiahol na ňom F1 skóre 98,69 %. O niečo nižší ale tiež vysoký detekčný výkon (F1-skóre 96,7 %) sa dosiahol aj na základe reprezentácií zreťazeného textu získaných pomocou XLNet [9] a Latent Dirichlet Allocation (LDA) v doprednej neurónovej sieti [10]. Bližší pohľad na ostatné prístupy riešenia zdieľanej úlohy CONSTRAINT sú vysvetlené v priložených článkoch.

Experimenty vykonané na súbore údajov FakeCovid používali vopred pripravený model BERT na reprezentáciu textového obsahu anglických spravodajských článkov z FakeCovid a získali skóre F1 0,78 [4].

Justus Mattern a kolektív vytvorili rozsiahli dataset FANG-COVID v nemeckom jazyku, na ktorom uskutočnili radu experimentov. Vzhľadom k charakteru datasetu, ktorý obsahuje ako textové dáta tak aj sociálnu interakciu boli tieto experimenty zamerané na vplyv použitých vlastností. Najlepšie výsledky dosiahol model BERT plus sociálny kontext [11], ktorý využíva ako textové dáta tak aj informácie o tvorcovi obsahu. Z výsledkov najdôležitejších vlastnosti vyplynulo, že oveľa dôležitejší je tvorca správy ako samotný text. To potvrdzuje, že ľudia s istým zmýšľaním majú vyššiu tendenciu šíriť nepravdivé správy.

Cui a Lee uskutočnili experimenty so súborom údajov CoAID pomocou rôznych klasifikačných modelov [5], ktoré sa ukázali ako úspešné pri všeobecnej detekcii falošných správ. Najmä niektoré z najvýkonnejších modelov na všeobecných súboroch údajov na detekciu falošných správ, ako sú CSI [12] a SAME [5], sa ukázali ako menej účinné pri CoAID, pričom dosiahli skóre F1 0,228. a 0,340 v porovnaní s najvýkonnejším modelom dEFEND so skóre F1 0,581 [13].

Pre slovenčinu sa v oblasti covid-19 fake-news zatiaľ žiaden hlbší výskum nevykonal. Sarnovský a kolektív publikovali dataset. Súčasťou publikácie bol iba základný výskum v oblasti klasifikácie fake-news v slovenskom jazyku [2]. Zistenia, že všeobecné modely pre angličtinu nefungujú na tejto špecifickej téme fake-news a nedostatok experimetov pre iné jazyky, len ďalej dokazujú potrebu výskumu detekcie falošných správ špecifických kritických udalostí, akou je pandémia.

2. Fake-news

Táto kapitola poskytuje stručný úvod do problematiky Fake-news, ktorou sa táto práca zaoberá.

Obr. 2.1: Vývoj vyhľadávania termínov súvisiacich s fake-news pomocou Google search za obdobie 1.5.2014 až súčastnosť. Predošlé obdobie je skoro identické prvej tretine grafu.

Termín „Fake-News“ sa začal často používať po voľbách v USA v roku 2016 [14], nárast je pekne viditeľný na obrázku 2.1, ktorý zobrazuje množstvo dotazov na google vyhľadávač v téme fake-news. Dve samostatné vyšetrovania Guardian a BuzzFeed identifikovali najmenej 140 webových stránok produkujúcich falošné správy ohľadom volieb zamerané na občanov USA, pričom všetky boli z malého mesta v Macedónsku.

Najvyššiu hodnotu dosiahol tento fenomén v marci 2020, kedy prebiehala pandémia covid-19. V tejto už tak ťažkej dobe sa začali šíriť dezinformácie a fake-news ešte rýchlejším tempom. Predpokladaná príčina spočívala v tom, že zúfalí ľudia v tomto psychicky vypätom období sú nachylnejší na deziformovanie. Po ustálení situácie s pandémiou sa znižuje množstvo fake-news. Ale ako môžeme vidieť na priebehu grafu, znovu narastajú, čo pravdepodobne bude mať súvis s ďalšou kritickou situáciou, a tou je vojna na Ukrajine. Priamy súvis medzi fake-news a kritickými situáciami vo svete nie je zatiaľ dokázaný. Ale fakty nasvedčujú tomu, že to v blízky budúcnosti bude ukázané.

Fake-news robia problémy aj v kľudných dobách. V takýchto kritických situáciách už sú životu nebezpečné, pretože ľudia v psychickom vypätí strácajú kritické myslenie a sú náchylnejší k oklamaniu pomocou fake-news.

Preto sme sa v tejto práci rozhodli zamerať práve na fake-news z obdobia covid-19.

2.1 Informačná bublina

Ako sme už spomenuli v úvode úzko súvisiacim problémom spojeným s termínom fake-news je fenomén informačnej bubliny. Informačná bublina je označenie pre výsledok tzv. personalizovaného vyhľadávania. Pri takejto personalizácii obsahu, algoritmy webových stránok filtrujú informácie s cieľom zobraziť užívateľovi len tie, ktoré chce vidieť (takže predovšetkým tie, ktoré korešpondujú s jeho názormi) zatiaľ, čo od ostatných informácií ho to izoluje.Tieto algoritmy vychádzajú z vyhodnotenia predchádzajúceho správanie užívateľa na internete a sú používané napríklad stránkami Facebook, Yahoo News alebo Google Search [15].

Personalizovanie produktov pre užívateľov sa javí ako prospešné. Ale ak dôjde k personalizácii diskusných príspevkov alebo politických, náboženských, sociálnych a mnohých iných názorov. V kontexte ak si užívateľ neuvedomuje, že informácie s ktorými prichádza do kontaktu sú filtrované. Môže takéto personalizovanie viesť na radikalizáciu názorov. Pretože používateľa izoluje od informácií, ktoré by u neho mohli viesť k novému zhodnoteniu problému a väčšej názorovej diverzifikácii. Personalizácia vyhľadávania naopak používateľa ešte hlbšie utvrdzuje v jeho presvedčení. Dôsledkom tohto javu môže byť dezinformovanosť a manipulovateľnosť užívateľov pomocou fake-news [16].

Na záver musíme ešte podotknúť, že situácia sa zlepšovať nebude. Ak s tým niečo neurobíme. Ústup klasických medií a nástup šírenia informácií pomocou internetu je nezastavitelný. Internet prekonal v roku 2018 všetky ostatné media v množstve šírenych nových informácií. Preto je v dnešnej dobe otázka správnej a rýchlej detekcie šírenia falošných správ ešte aktuálnejšie ako kedykoľvek predtým.

2.2 Boj Európskej únie proti dezinformáciam

Európska únia si je vedomá sily šírenia dezinformácií, a preto má zriadenú diplomatickú službu East StratCom Task Force. V roku 2015, bolo v rámci tejto diplomatickej služby vytvorený tím, ktorý má za úlohu sa zaoberať prokremeľskými prebiehajúcimi dezinformačnými kampaňami. A ich vplyvom na politické zriadenia v Europskej únii.

East StratCom Task Force zverejnil analýzu s názvom East StratCom Task Force The legal framework to address "fake-news": possible policy actions at the EU level [1]. Práca popisuje šírenie dezinformácií a sumarizuje kľúčové zistenia. Vrátane klesajúcej dôvery v internet, nízkej schopnosti jeho užívateľov vyčleniť pravdivé správy hodné ich pozornosti. Ďalšími zisteniami boli, že mnohé falošné a manipulatívne správy sú ignorované a stále niektorá časť z nich existuje na internete. A tým sa opätovne môže rýchlo šíriť a ovplyvňovať verejnú mienku v Európskej únii. Pretože tým vytvára „šum", že poskytuje protichodné informácie k jednej udalosti. Čo v konečnom dôsledku spôsobuje polarizáciu spoločnosti a zmätok pri rozhodovaní.

2.3 Fake-news na Slovensku

Falošné správy, tiež známe ako "fake-news", sú vážnym problémom na Slovensku, rovnako ako aj v mnohých iných krajinách sveta [2].

Na Slovensku, je tento problém o to viac spôsobený aktivitov krajne pravicovej politickej scény, ktorá systematicky šíri dezinformácie napríklad o EÚ alebo NATO. V prvej vlne covid-19 sa zamerali najmä na dezinformácie o tom, že rómovia majú rôzne výhody a že sa o nich štát stará viac aj počas pandémie. Takto sa snažili viac polarizovať spoločnosť. Snažili sa prezentovať manipulácie s číslami nakazený až smrteľnosť vakcinácie. Tým chceli zvýšiť nedôveru v klasické media, ktoré poukazujú na ich krajne pravicové postoje. A to všetko len pre zvýšenie popularity a zlepšenie verejnej mienky [17].

Samostatnou skupinou sú platení šíritelia falošných správ na Slovensku, títo ľudia neváhajú šíriť akékoľvek informácie, plus ich propagovať, ak za to majú zaplatené. Na Slovensku sa niektoré označujú za "relevantné media". O to je to väčší problém na tak malú krajinu ich je priveľa.

Aj na Slovensku už začínajú projekty ohľadom osväty v témach falošných správ, ale zatiaľ sú to len začiatky. Automatické systémy na fact-checking zatiaľ neexsitujú a fyzickým Facebook fact-checkerom na Slovensku je jedná osoba, ktorá má právo označiť na tejto platforme hoax alebo falošné správy.

Aj z toho dôvodu sú potrebné automatické systémy určené na detekciu fake-news pre slovenské texty.

2.4 Definícia deziformácie

Cieľom tejto práce nie je riešiť filozofickú rovinu definície fake-news. Ale pre lepšie pochopenie problému, ktorý sa snažíme vyriešiť pomocou NLP je vhodná definícia základného problému. Podľa definície New York Times sú fake-news správy, ktoré vznikli so zámerom oklamať čitateľa. Na druhú sranu tím EÚ, ktorý bojuje proti šírenie dezinformácii v európskej únii má zadefinované pod týmto pojmom správy s jasným zámerom manipulovať verejnú mienku [1].

V tejto práci nebudeme skúmať úmysel s akým bola správa vytvorená, či vznikli omylom alebo zámerne. Preto definujeme všetky nepravdivé správy ako fake-news.

3. Detekcia Fake-news

Detekciu falošných správ a dezinformácií šírených na internete je možné rozdeliť na niekoľko podproblémov, ktoré je možné riešiť samostatne alebo ako ich kombináciu. V tejto kapitole popíšeme možnosti automatizácie riešenia jednotlivých podproblémov, ktoré už boli vykonané. Na druhú stranu popíšeme aj možnosť využitia strojového učenia a neurónových sieti na riešenie týchto čiastkových problémov. Výsledne klasifikačné skóre by bolo určené ako kobinácia výsledkov riešení týchto čiastkových problémov. Tento prístup minimalizuje riziko falošne pozitívnej klasifikácie.

3.1 Non-NLP metódy

V tejto práci sa zaoberaméme detekciou falošných správ vzhľadom k ich textovému obsahu. Teda tento problém chápeme ako úlohu textovej klasifikácie. tento princíp detekcie falošných správ vyplynul z dostupných dát pre slovenský jazyk. Ale hlavná motivácia spočíva v cieli zistiť, či je možné falošné správy detekovať len na základe špecifík použitého jazyka.

Na druhú stranu existuje viac prístupov k detekcii falošných správ. Niektoré z princípov ako detekovať falošné správy, ktoré už boli preskúmane, stručne zhrnieme v nasledujúcich podkapitolách.

3.1.1 Overenie zdroja

Pravdivé spravodajské články by mali vždy obsahovať meno autora a zdroj. Väčšina nepravdivých správ tento zdroj neobsahuje. Ale niektoré stránky sa to naučili obchádzať tým, že vytvárajú reťazec mnohých článkov publikovaných na rôznych webových stránkach, aby navodili dojem, že informácie v článku sú dostatočné pokryté zdrojmi. No v skutočnosti reálny zdroj chýba [1].

Automatická kontrola zdrojov by mohla byť vítanou funkciou, ktorá by odhalila pôvod informácie. A teda či pochádza z hodnoverného zdroja alebo ak by sa v reťazci zdrojov zacyklila označila by danú informáciu za nedôverihodnú. Výsledne hodnotenie by bolo zahrnuté v celkovom hodnotení všetkých podproblémov. Netreba pri takomto overovaní zdrojov zabudnúť na obrázky a videá, ktoré sú súčasťou článku.

3.1.2 Stance Detection

Úlohou Stance Detection je odhadnúť relatívny pohľad (alebo postoj) dvoch častí textu vzhľadom na tému, tvrdenie alebo problém. Bolo to považované za užitočné v súťaži „Fake News Challenge“, ktorá sa snažila použiť AI techniky na detekciu dôveryhodnosti článku. Tiež navrhli rozdelenie procesu kontroly do niekoľkých etáp. Stance Detection bol cieľom prvej etapy.

Presnejšie, predstavili nasledujúce triedy pre detekciu postoju:

- Súhlasí: telo textu súhlasí s titulkom
- Nesúhlasí: telo textu nesúhlasí s titulkom

- Diskutuje: telo textu diskutuje o rovnakej téme ako titulok, ale neprijíma postoj
- Nesúvisí: telo textu diskutuje o inej téme ako titulok

Počas súťaže bola taktiež navrhnutá metóda hodnotenia, ktorá pozostávala z dvoch krokov. Pvým krokom bolo uverenie súvislosti tela textu s jeho nadpisom. K celkomvému hodnoteniu to prispelo s váhou 1/4.

Druhý krok pozostával z určenia medzi pármi, ktoré spolú súvisia.Tím zo Stanfordu vykonal radu experimentov založených na nasledujúcich architektúrach.

- Concat MLP
- BoW MLP
- Dual GRU
- Bi-dir MSTL

Výskumníci so Stenfordu vykonali viac ako 100 experimentov na vylepšenie svojich modelov, ale iba BoW MLP dosiahlo lepšie skóre ako základné riešenie. Tento model dosiahol 0,97 presnosť v prvej podúlohe (súvisiaca alebo nesúvisiaca hlavička), 0,93 na druhej úrovni (klasifikácia súvisiacich párov) a 0,89 na celkovom NFC-1 skóre - čo znamená zlepšenie o 10% oproti základnému riešeniu.

Na druhú stranu, prekvapivo žiaden z modelov hlbokých neurónových sieti, založených na architektúre RNN nebol schopný prekonať základné riešenie. Vzhľadom k týmto výsledkom, uviedli v svojej práci možné vysvetlenia, prečo modely architektúry RNN dosahujú nižšie skóre ako base model. Aj keď sa predpokladalo, že budú viditeľne presnejšie.

Prvým možným dôvodom, ktorý uvideli: pretože dosahovali vysokú presnosť na trénovacích dátach, ale nikdy na dátach na validáciu, predpokladajú, že RNN by potreboval oveľa väčší dátový set na generalizáciu.

Ďalší možný dôvod bol, že aj pokročilé RNN modely - Bi-dir MSTL a Dual GRU - majú ťažkosti so zabúdaním dlhodobých súvislosti, ktoré môžu byť dôležité pri probléme klasifikácie článku. V závere svojej práce uvideli, že uvedené problémy chcú vyriešiť a zlepšiť svoje modely pre detekciu fake-news.

3.1.3 DeepFake

Termín Deepfake sa používa na označenie médií (zvyčajne audio alebo video), ktoré sú manipulované pomocou techník umelej inteligencie, konkrétne algoritmami strojvého učenia, na vytvorenie falošnej verzie pôvodného obsahu [1].

Potom jednou z možností ako odhaliť falošnú správu je skontrolovať obrázky priložené k článku. Kontrolou sa overí ich súvislosť s textom ale aj korektnosť jednotlivých priložených obrázkov. Cielene nesprávne zvolený obrázok môže manipulovať rovnako ako klamlivý nadpis alebo časť textu.

Častokrát falošné správy používajú obrázky, ktoré vznikli pri inej udalosti alebo boli upravené tak aby zapadali do falošnej témy danej správy ale nebolo ich možné odlíšiť od autentického obrázka, tak aby potvrdili týmto vizuálnym

obsahom informácie v texte. Teda hlavný princíp tohto problému spočíva vo využití audiovizuálneho obsahu najčastejšie obrázkov ako "dôkazového materiálu", s cieľom zvýšiť hodnovernosť zdieľanej falošnej správy [1].

Najčastejší spôsob ako určiť, či daný obrázok súvisí s textom je použitie neurónových sietí založených na architekúre konvolučných neuronových sieti [18]. Detekcia falošných obrázkov sa týka procesu identifikácie a klasifikácie obrázkov, ktoré boli manipulované alebo vygenerované umelo. Môže to zahŕňať identifikáciu deepfake obrázkov, obrázkov, ktoré boli upravené alebo pozmenené a obrázky, ktoré boli vygenerované degeneratívnymi modelmi, ako sú GAN.

3.2 Klasifikácia na základe textových dát

Klasifikácia falošných správ založená na textových dátach sa zaoberá triedením a kategorizáciou rôznych textových zdrojov na základe toho, či obsahujú nepravdivé informácie alebo manipulatívne techniky, ktoré môžu zaviesť čitateľa. Strojové učenie a algoritmy sa používajú na extrakciu príznakov z textu, ktoré následne pomáhajú určiť, či ide o falošnú správu alebo nie. Existuje viacero prístupov k tejto klasifikácii, napríklad štatistické metódy alebo algoritmy založené na hlbokom učení. V súčasnosti sa vyvíjajú nové nástroje a aplikácie, ktoré umožňujú rýchle a účinné klasifikovanie falošných správ na základe ich obsahu.

V tejto práci sa bližšie zameriame práve na tieto techniky detekcie falošných správ a ich aplikáciu na dataset obsahujúci texty v slovenskom jazyku.

3.2.1 Falošné správy generované pomocou NLP

Falošné správy generované neurónovými sieťami sú v tejto oblasti pomerne novinkou. Generovanie falošných správ pomocou techník spracovania prirodzeného jazyka (NLP) zahŕňa použitie algoritmov strojového učenia a neurónových sietí na vytváranie textu, ktorý je ťažké rozlíšiť od textu napísaného človekom. Tieto algoritmy dokážu analyzovať existujúci text a vytvárať nový text, ktorý sa používa na šírenie falošných alebo neoverených informácií.

Štúdia, ktorú Zellers a kolektív uverejnili, predstavuje model s názvom Grover na tvorbu kontrolovateľných spravodajských článkov [19]. Dokáže vygenerovať celý článok len na základe malého kúsku textu alebo nadpisu. Navyše, ak je vo vstupe uvedená doména spravodajského webu, vygeneruje sa text v štýle týchto novín. Neurónové falošné správy generované týmto modelom boli ľuďmi hodnotené ako dôveryhodnejšie ako správy písané ľuďmi.

Podobný výsledok je možné dosiahnuť aj pomocou modelu GPT-2, pri zadaní správneho dotazu [20]. Aj keď tento model na to nie je priamo určený. Spoločnosť OpenAI, už vydala jeho vylepšenú verziu GPT-3, ktorá ma aj verejne dostupného chatbota, čo prácu s týmto modelom ešte uľahčuje. Pomocou tohto verejne dostupného chatbotu vie generovať text aj laik [21].

Generovanie falošných správ pomocou NLP môže byť použité na rôzne účely, ako napríklad šírenie dezinformácií alebo manipulácia verejnou mienkou. Takéto generovanie falošných správ je o to vážnejšie, že sa môže diať automaticky a plne individuálne prispôsobovať každému konečnému čitateľovi na základe jeho preferencii.

4. Príprava vstupných dát

V tejto podkapitole sa budeme venovať úprave získaných vstupných textových dát. Tento krok predchádza samotnej klasifikácii. Vo všeobecnosti platí, že ak sa pozrieme z časového hľadiska na celý proces od získania vstupných dát až po ich klasifikáciu pomocou modelu strojového učenia. Potom príprava vstupu do vhodného formátu tvorí najväčšiu časť tohto procesu.

Vzhľadom na to, že spracovávané dáta môžu mať veľmi odlišný formát, vykonáva sa v tejto fáze normalizácia textu. Ide o proces, ktorým sa textový vstup upraví tak, aby bol vhodný pre automatické strojové spracovanie. Počas tohto kroku ako sám názov napovedá dochádza k štandardizácii vstupu do jednotného formátu. Normalizácia najčastejšie pozostáva z týchto krokov:

- Zmena formátu písmen: Všetky písmená sa môžu upraviť na malé alebo veľké, nezáleží na akú veľkosť, nutné je len aby sa štandardizovala veľkosť písmen v celom texte.
- Tokenizácia: Proces, ktorý rozdeľuje text na jednotlivé skupiny znakov, oddelené medzerami alebo inými symbolmi. Tieto skupiny znakov sa nazývajú tokeny a môžu reprezentovať slová, frázy alebo dokonca celé vety [22].
- Odstránenie interpunkcie: Interpunkčné znamienka ako bodky, čiarky alebo dvojbodky sa môžu odstrániť, aby sa zjednotil formát textu. Pretože interpunkcia nemá žiaden špecifický význam pre následne analyzovanie textu, no jej odstránenie nám pomôže skrátiť dĺžku textov.
- Odstránenie diakritiky: Diakritické znamienka (napr. čiarky, bodky, háčiky) na písmenách sa môžu odstrániť, aby sa zjednotila forma slov a výrazov.
- Odstránenie stop slov: Slová ako „a“, „alebo“, „na“, „pri“, „s“ atď. sa môžu odstrániť, aby sa zredukoval počet slov v texte a zlepšila sa kvalita analýzy [22].
- Lemmatizácia: Proces je založený na odstraňovaní flexie slova pomocou rôznych pravidiel a tým upraviť slovo na základny tvar, tzn. slovo „bežať“ sa zjednotí s tvarmi „bežím“, „bežal“, „bežal by som“ na základný tvar „bežať“. No s dôrazom na to, že výsledkom lematizácie je skutočné slovo t.j. léma. Léma je základný tvar slova, slovníkový tvar. To túto metódu odlišuje napríklad od izolácie koreňa slova, kde výsledkom nie je skutočné slovo [23].

Tento proces štandardizácie dát, ktorý v sebe zahŕňa odstránenie diakritiky, stopwords, interpunkcie a ďalších znakov jazyka sme využili pre prípravu dát pre základne modely strojového učenia. Pre modely založené na architektúre Transformer sme vstupné dáta očistili iba od netextových komponent.

Normalizácia vstupu je veľmi dôležitým krokom pre ďalšie procesy úpravy dát. Ak by sme nespracovali text do normalizovanje formy, teda jednoduchšej na spracovanie pre počítač. Potom by toto automatické spracovanie počítačom bolo náročné, a preto sú tieto úpravy nevyhnutné [22].

Po tomto kroku získavame text očistený od šumu a nepotrebných slov, z ktorých nie je možné získať dodatočné informácie. ďalším benefitom takéhoto spracovania dát je skrátenie vstupných reťazcov, čo má za následok zrýchlenie nasledujúcich krokov [23].

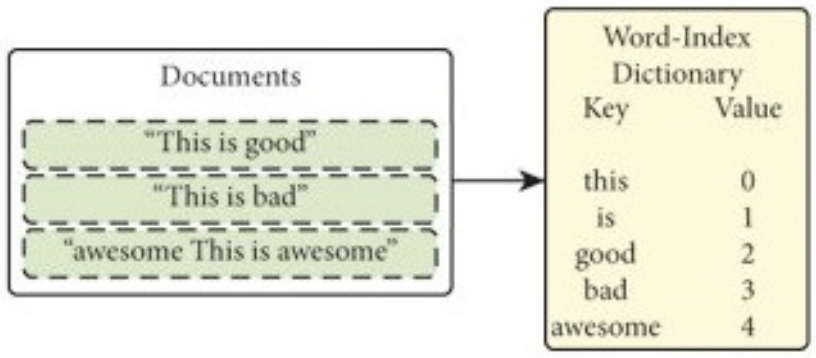
4.1 Vektorizácia textu

Po úprave textových dát pomocou predprípravy máme k dispozícii dátový súbor vo vhodnej forme. Ďalším krokom je transformácia týchto dát do numerického formátu, v podobe vektoru čísel, ktorý reprezentuje vstupný textový dokument. Tento proces sa nazýva vektorizácia vstupu. vykonanie tohto procesu je nevyhnutné, pretože väčšina algoritmov strojového učenia pracuje s numerickými dátami a preto je nutné textové dáta transformovať do vektorovej formy.

Existujú viaceré druhy spôsobov transformácie textových dát do vektorového kódovania. V tejto práci sa budeme venovať transformácií textových dát do numerického priestoru za pomoci dvoch z nich, a to:

- Frekvencie tokenov
- TF–IDF

Pre ilustráciu, ako fungujú jednotlivé vektorizačné modely, si vytvoríme jednoduchý korpus, ktorý bude pozostávať z troch dokumentov. Obrázok 4.1 znázorňuje vytvorenie slovníka na zvolenom korpuse zo všetkých unikátnych tokenov v jednotlivých vetách.



Obr. 4.1: Slovná zásoba korpusu [24].

V každom korpuse existuje množstvo jedinečných tokenov, ktoré sú od seba oddelené a spolu tvoria slovnú zásobu tohto jazykového korpusu. Táto slovná zásoba je tvorená jednotlivými tokenmi, ktoré sa nachádzajú v textoch, pričom opakujúce sa slová sú v slovníku zobrazené iba raz. Pre lepšiu orientáciu je vytvorený slovník zoradený abecedne.

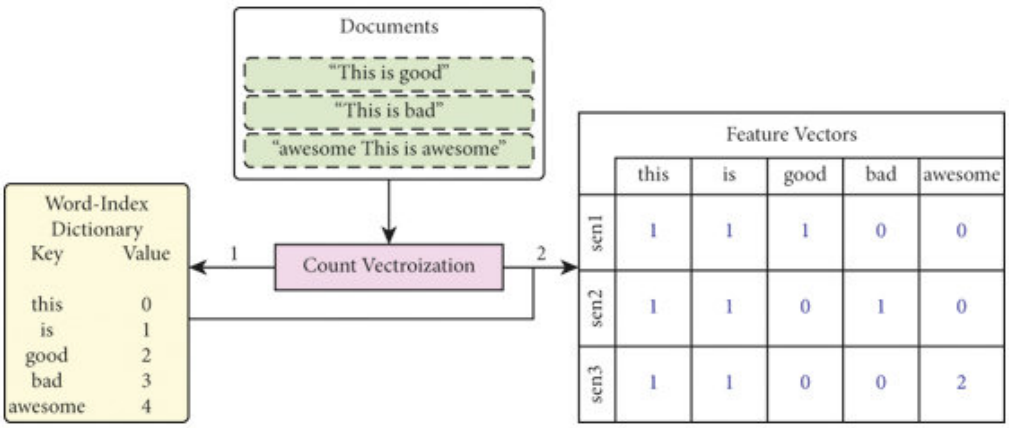
4.1.1 Frekvencia tokenov

Veľmi efektívnym a zároveň najjednoduchším modelom, ktorý sa využíva na transformáciu vstupu je Bag of words. Častejšie uvádzaný pod skratkou BoW. Majme súbor textov, ktorý sa v úlohách spracovania prirodzeného jazyka nazýva korpus. Potom termín frekvencia tokenov môže byť charakterizovaný ako počet jednotlivých tokenov v danom texte [24].

Hlavná myšlienka celej transformácie je veľmi jednoduchá, ale zároveň účinná. Touto myšlienkou je určenie výskytu slov v dokumente, pričom sa ignoruje ich poradie v texte. Vysoký počet niektorého z tokenov, značí jeho dôležitosť pri rozhodovaní [24].

Ak aplikujeme tento porces na každý text v korpuse, potom sú tieto texty reprezentované pomocou vektora čísel s fixnou dĺžkou. Dĺžka tohto vektora je závislá na veľkosti slovníka korpusu.

Princíp fungovania tohto algoritmu si ukážeme na jednoduchom korpuse, ktorý sme si zadefinavali v úvode tejto sekcie. Tieto vety budeme vektorizovať pomocou vyššie popísaného algoritmu. Jeden z možných postupov vektorizácia vstupných dokumentov vidíme nižšie.



Obr. 4.2: Schéma algoritmu frekvencie slov [24].

V pravej časti ilustrácie 4.2 je umiestnená výsledná tabuľka, ktorá zobrazuje vektorizovaný korpus pomocou algortimu BoW. V prvom riadku môžeme vidieť, či sú slová z vytvoreného slovníka prítomné alebo chýbajú v prvom texte. Podobne druhý riadok zobrazuje prítomnosť a frekvenciu slov v druhom texte a takto môžeme pokračovať pre všetky texty v korpuse. Stĺpce tabuľky zodpovedajú slovám zo slovníka reprezentujúceho celú slovnú zásobu daného korpusu. Ich počet je daný veľkosťou slovníka. Slová v slovníku respektíve stĺpce sú v našom prípade zoradené v zostupnom abecednom poradí.

Z ilustrácie 4.2 pre náš konkrétny kropus je zrejmé, že slovník pre tento korpus je tvorený 5 unikátnymi slovami z troch textov. Prvým slovom v slovníku je „this“, ktoré má index 0, a posledným slovom „awesome“ s indexom 4.

V tejto práci sme pre vektoriyáciu vstupu pomocou metódy BoW, použili implementáciu CountVectorizer() v *sklearn* knižnici v jazyku python, ktorú si je možné prisbôsobiť rôznou voľbou vstupných parametrov.

4.1.2 TF–IDF

Druhou metódou kódovania, ktorú si uvedieme, je TF–IDF (z angl. Term Frequency – Inverse Document Frequency). Táto metóda je oveľa komplexnejšia a vo svojej vektorovej reprezentácii zohľadňuje aké je dané slovo dôležité vzhľadom k celému súboru textov [24].

Transformácia slova na vektor pozostáva z dvoch primárnych častí - frekvencie termov (TF) a inverznej dokumentovej frekvencie (IDF). TF udáva počet výskytov slova v konkrétnom texte a teda vyjadruje jeho dôležitosť v rámci tohto textu. IDF nám na druhej strane poskytuje informáciu o tom, ako často sa dané slovo vyskytuje v iných textoch. Ak je slovo dôležité v jednom texte, nemalo by sa vyskytovať príliš často v iných textoch. Táto informácia sa používa na váhovanie významu slova v rámci celej kolekcie textov, nie iba v rámci jedného textu.

TF-IDF sa vypočíta pomocou vzorca [24]:

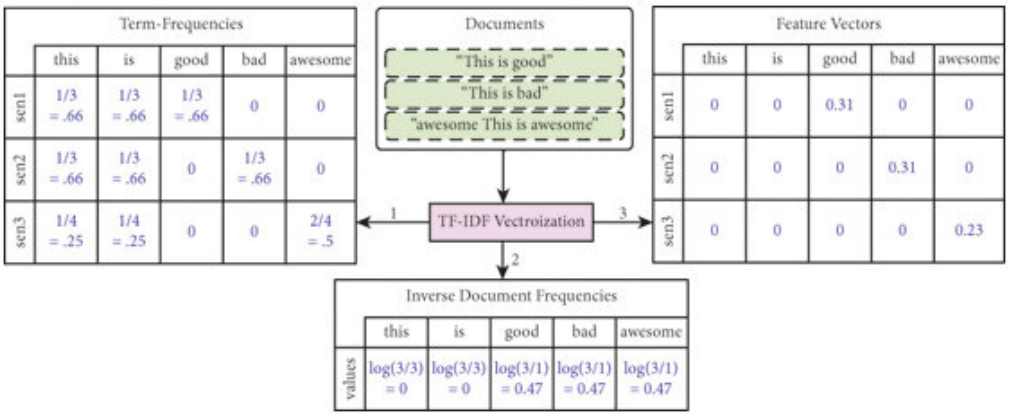
$$TFIDF = TF \times IDF \tag{4.1}$$

štandardná definícia TF je

$$TF(t,d) = \frac{frekvencia\ tokenu\ t\ v\ dokumente\ d}{celkový\ počet\ termov\ v\ dokumente} \tag{4.2}$$

a IDF vyjadríme pomocou vzorca

$$IDF(t,D) = \log \frac{celkový\ počet\ dokumentov\ v\ datasete}{počet\ dokumentov,\ ktoré\ obsahujú\ token\ t} \tag{4.3}$$



Obr. 4.3: Schéma algoritmu TF-IDF [24].

Na ilustrácii 4.3 je zobrazený výpočet podľa štandardnej definície TF-IDF v nami uvedenom korpuse.

V tejto práci je však na transformáciu vstupných textov použitá implementácia v knižnici *sklearn* v programovacom v jazyku python. Podobne ako pri Bag of words a funkcii CounTVectorizer aj túto si môžeme prispôsobiť rôznou voľbou vstupných parametrov, ktoré sú na výber rovnako ako pri predošlej metóde vektorizácie vstupu. [25]. Výstup z funkcie je štandardne automaticky normalizovaný.

Vzorec pre výpočet TF-IDF sa môže v rôznych implementáciach vyskytovať s drobnými obmenami, je tomu tak aj v tejto knižnici, ktorá z nutnosti výpočtov na počítači mierne túto definiciu upravila do nas,edovnej formulácie:

$$TFIDF = TF \times IDF \tag{4.4}$$

kde TF je vyjadrené ako

$$TF(t,d) = frekvencia\ tokenu\ t\ v\ dokumente\ d \tag{4.5}$$

a IDF vyjadríme pomocou vzorca

$$IDF(t,D) = \ln \frac{celkový\ počet\ dokumentov\ v\ datasete\ +\ 1}{počet\ dokumentov,\ ktoré\ obsahujú\ token\ t\ +\ 1} \tag{4.6}$$

Úprava pôvodnej definície v IDF je spôsobená tým [24], že ak by slovníkové slovo neobsahoval, žiaden z textov potom by mohlo dôjsť k deleniu nulou a spadnutiu celého modelu. Nato nadväzuje aj pripočítanie jednotky v čitateli. Ďalej táto definícia, ktorú sme si uviedli pripočítava k logaritmu +1, a to z toho dôvodu ak by bol logaritmus nulový, teda ak by sa dané slovo nachádzalo vo všetkých textoch.

V tejto sekcii sme si ukázali dva spôsoby vektorizácie textu, a to BoW a TF-IDF. Prvým algoritmom bol BoW, udáva frekvenciu slov t.j. koľkokrát sa dané slovo nachádza v danom texte. Je veľmi jednoduchý a prehľadný.

Na druhú stranu TF-IDF berie do úvahy aj váhu slova vzhľadom k celému korpusu. Zohľadňuje aj aké dôležité je konkrétne slovo v danom texte, resp. v celom súbore texte.

5. Vyhodnocovanie výsledkov

Pre porovnanie odlišných metód strojového učenia a vyhodnotenie ich úspešnosti sa používajú rôzne metriky. Značné množstvo metrík, ktoré sú určené pre klasifikačné úlohy, sú založené na hodnotách uvedených v konfúznej matici. [26]. Tieto metriky si priblížime v tejto kapitole.

5.1 Konfúzna matica

Zvyčajne sa na prvotnú analýzu výsledkov klasifikačných metód využíva konfúzna matica. Táto matica sa často používa na vypočítanie ďalších komplexnejších metrik, ktoré sú založené na hodnotách v tejto matici.

		Trieda predikcie	
		Positive	**Negative**
Skutočná trieda	**Positive**	True Positive (TP)	False Negative (FN)
	Negative	False Positive (FP)	True Negative (TN)

Obr. 5.1: Konfúzna matica pre binárnu úlohu

Na ilustrácii 5.1 je znázornená konfúzna matica pre binárnu klasifikačnú úlohu t.j. konfúzna matica dvoch tried. Potom riadky matice reprezentujú skutočné triedy a stĺpce predstavujú predikované triedy príkladov, prípadne môže byť matica transponovaná. Matica znázornená na ilustrácii 5.1, má hlavnú diagonálu, ktorá zodpovedá správnym predikciám, a vedľajšiu diagonálu, ktorá zodpovedá tým nesprávnym.

V prípade, že pozitívne príklady sú označené hodnotou 1 a negatívne príklady hodnotou 0, kategória *True Positive* (TP) vyjadruje počet pozitívnych príkladov, ktoré boli správne klasifikované. Podobne, kategória *True Negative* (TN) vyjadruje počet správne klasifikovaných negatívnych príkladov. Potom kategória *False Negative* (FN) zahrňuje pozitívne príklady, ktoré boli nesprávne klasifikované ako negatívne príklady. Na druhej strane, kategória *False Positive* (FP) označuje počet príkladov, ktoré sú negatívne, ale boli nesprávne klasifikované ako pozitívne príklady. Tieto dve kategórie sú poslednými kategóriami pre binárnu klasifikačnú úlohu [27].

5.2 Accuracy

Táto metrika vyjadruje pomer počtu správnych predpovedí k celkovému počtu vstupov. Výpočet tejto metriky na základe hodnôt konfúznej matice je matematicky vyjadrený rovnicou 5.1.

$$acc = \frac{počet\ správnych\ predikcií}{celkový\ počet\ predikcií} \tag{5.1}$$

V prípade rapídne nevybalansovaných klasifikačných tried v datasete nie je vhodné použiť metriku Accuracy na vyhodnotenie kvality klasifikačného modelu. V takejto situácii táto metrika ťažko odhaľuje nesprávnu klasifikácii triedy s menším počtom vstupných vzorov.

5.3 Precision a recall

Podobne ako pri accuracy sa aj hodnota precision vypočíta z konfúznej matice a určuje pomer správne pozitívne predikovaných príkladov ku všetkým, ktoré boli klasifikované ako pozitívne. Tento vzťah sa dá vyjadriť pomocou rovnice 5.2.

$$precision = \frac{TP}{FP + TP} \tag{5.2}$$

Recall vyjadruje percento príkladov danej triedy, ktoré klasifikátor správne identifikuje. Táto metrika indikuje, ako dobre model dokáže rozlišovať jednotlivé triedy. V prípade, že má recall hodnotu 1, klasifikátor dokáže danú triedu identifikovať bezchybne.

$$recall = \frac{TP}{FN + TP} \tag{5.3}$$

Vysokú hodnotu precision je možné dosiahnuť tiež spôsobom, že klasifikačná metóda vráti iba jeden pozitívny prípad, pričom ešte existuje veľké množstvo pozitívnych t.j. anotovaných vzorov s hodnotou 1. Takáto situácia je však nežiadúca a hodnota metriky recall bude v tomto prípade veľmi nízka. Na druhej strane opačným extrémom je, keď klasifikátor predikuje všetko ako pozitívne. V takejto situácii je hodnota recall veľmi vysoká ale typicky potom bude málo zo všetkých pozitívne predikovaných dokumentov relevantných, čo tiež nie je štandardný stav. Na druhú stranu v takomto prípade bude hodnota precision veľmi nízka.

Preto sa tieto metriky najčastejšie vyjadrujú v podobe závislosti hodnoty precision a recall napríklad v podobe precision-recall kriviek. Nevýhodou týchto metrík je, že sú dve a následne porovnanie rôznych klasifikačných metód je veľmi náročné. Preto boli navrhnuté aj iné metriky, ktoré v svojom výpočte priamo kombinujú tieto dve metriky.

5.4 F1 skóre

Metrika F1 skóre vyjadruje harmonický priemer medzi hodnotami precision a recall. Pre najlepší výsledok dosahuje hodnoty 1, naopak pre najhorší 0 [28]. V prípade klasifikácie viacerých tried sa táto metrika počíta zvlášť a nakoniec je spriemerovaná do $F1_{macro}$.

$$F_\beta = (1 + \beta^2) \cdot \frac{precision \cdot recall}{(\beta^2 \cdot precision) + recall} \tag{5.4}$$

Ak je $\beta > 1$, potom je kladený väčší dôraz na recall, naopak ak je $\beta < 1$ je dôraz kladený na precision. Možnosť zmeny dôrazu má veľké množstvo výhod,

ale nesie so sebou aj jednu značnú nevýhodu. Ak sú pri vyhodnocovaní rôznych modelov použité odlišné voľné parametre β, potom je vzájomne porovnanie týchto modelov veľmi ťažké.

Hlavnou výhodou tejto metriky je to, že je kombináciou metrík a teda nemá sklony k takej chybovosti ako predošlé uvedené metriky. Ďalšou výhodou je možnosť zmeny dôrazu na precision alebo recall. Podľa toho, ktorá z metrík je pre nás dôležitejšia. Ak chceme dosiahnuť rovnakú váhu oboch metrík, teda $\beta = 1$:

$$F1 = 2 \cdot \frac{precision \cdot recall}{precision + recall} \tag{5.5}$$

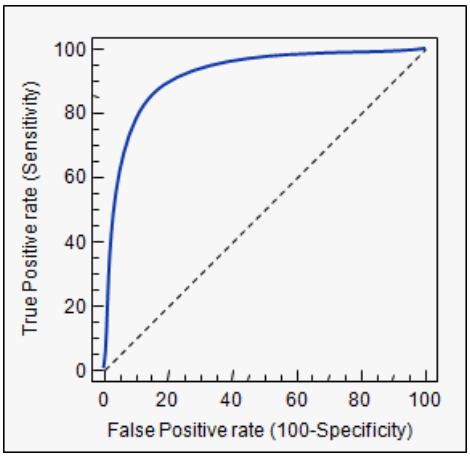
5.5 AUC-ROC krivka

Názov, ktorý sa používa pri tejto metrike je skratkou celého názvu v angličtine „Area Under the Receiver Operating Characteristics“, taktiež je zaužívaná skrata AUROC.

$$TPR = \frac{TP}{TP + FN} \tag{5.6}$$

$$FPR = \frac{TN}{FP + TN} \tag{5.7}$$

Krivka ROC je vytvorená na základe hodnôt *True Positive Rate* (TPR), ktorá reprezentuje hodnotu na osi y, a *False Positive Rate* (FPR), ktorá reprezentuje hodnotu na osi x.



Obr. 5.2: ROC krivka [29].

Metrika AUC-ROC určuje schopnosť klasifikátora rozlišovať medzi klasifikovanými triedami na základe hodnôt, ktoré sú v konfúznej matici. Úspešnosť klasifikátora sa vyhodnocuje na väčšo, množstve behov, pri ktorých sa menia učiace parametre totho algoritmu. Ak hodnota AUC-ROC je aproximovane 1, indikuje to, že klasifikátor lepšie rozlišuje jednotlivé triedy. Na druhú stranu, ak je hodnota blízka nule, znamená to, že klasifikátor má najhoršiu schopnosť rozlišovať triedy a predikuje ich v binárnej klasifikačnej úlohe opačne [30].

6. Strojové učenie

Táto kapitola bude venovaná klasifikacii textu pomocou rôznych modelov strojového učenia. Klasifikácia textu označuje proces, pri ktorom sú vstupné texty kategorizované do predom definovaných tried. Tento automatický proces spracovania textu sa využíva v rôznych oblastiach detekcie, ako napríklad:

- spamové filtre
- triedenie novinových článkov alebo e-mailov podľa obsahu
- detekcia falošných správ alebo podvodov

Klasifikátor ma za úlohu priradiť každý dokument do správnej klasifikačnej triedy. Ešte pred samotným tréningom klasifikačného modelu a jeho otestovaním je nutné v prvom kroku vstupný dátový súbor rozdeliť na trénovaciu a testovaciu časť. Až po vykonaní tohto kroku môžeme na trénovacích dátach vybudovať klasifikátor. A následne v ďalšom kroku takto naučený klasifikator otestovať na množine testovacích dát. Pri testovaní na overenie úspešnosti modelu používame rôzne metriky ako accuracy, f1-score, precision, recall a atď., ktoré sme popísali v kapitole 5.

Cieľom klasifikácie textových dát je rozdeliť množinu m samostatných dokumentov D_1, D_2,..., D_m obsiahnutých v množine textových dát D, ktorú tiež nazývame korpus, do n tried $T_1, T_2, ..., T_n$ na základe ich charakteristík. Klasifikátor sa snaží priradiť každý dokument D_i, pre $i = 1,2, ..., m$, do najvhodnejšej triedy T_j, pre $j = 1, 2, ...,n$, na základe podobnosti dokumentu s touto triedou. Nie je vždy možné jednoznačne priradiť dokument do konkrétnej triedy, ale klasifikátor sa snaží nájsť najlepšiu možnú zhodu, teda najpodobnejšiu triedu dokumentov, ktorým sa daný dokument najviac približuje, aj keď nie stopercentne.

Na klasifikáciu textu existuje viacero rôznych algoritmov strojového učenia. V nasledujúcich sekciách si niektoré z nich popíšeme, ktoré sme v tejto práci použili na klasifikáciu falošných správ v slovenskom jazyku.

6.1 Naivný Bayesov klasifikátor

Naivný Bayesov klasifikátor je založený na princípe pravdepodobnostného klasifikovania a využíva predpoklad nezávislosti vstupných dát. Oproti iným klasifikačným metódam, jeho hlavnou výhodou je, že nepotrebuje veľké množstvo trénovacích dát na učenie. Naopak, na určenie parametrov jednotlivých tried mu stačí relatívne malá vzorka trénovacích vzorov.

Tento model strojového učenia klasifikuje vstup na základe Bayesovej vety. Konkrétne v našej úlohe detekcie falošných správ, množinu textov. Znenie tejto vety je nasledovné: „Majme dve náhodne javy A a B s pravdepodobnosťami $P(A)$ a $P(B)$, kde $P(B) \neq 0$. Potom za predpokladu, že nastal náhodný jav B je možné podmienenú pravdepodobnosť javu A vyjadriť ako:“

$$P(A|B) = \frac{P(B|A) \cdot P(A)}{P(B)} \tag{6.1}$$

V predošlom odstavci sme uvideli definíciu Bayesovej vety a v tejto časti sa budeme venovať jej aplikácii v našej konkrétnej úlohe klasifikácie falošných správ na základe textového obsahu. Nech existujú cieľové klasifikačné triedy $T_1, T_2, ..., T_n$, potom úlohou Naivného Bayesovho klasifikátora je vypočítať podmienenú pravdepodobnosť, že nový dokument D_{new} s určitými charakteristikami $d_1, d_2,..., d_l$ patrí do triedy T_j pre $j = 1,2,...n$.

Túto pravdepodobnosť pre klasifikačnú triedu T_j vypočítame podľa nasledujúceho vzorca:

$$P(T_j|d_1, d_2,..., d_l) = \frac{P(d_1, d_2,..., d_l|T_j)P(T_j)}{P(d_1, d_2,..., d_l)}, \qquad 1 \leq j \leq n \tag{6.2}$$

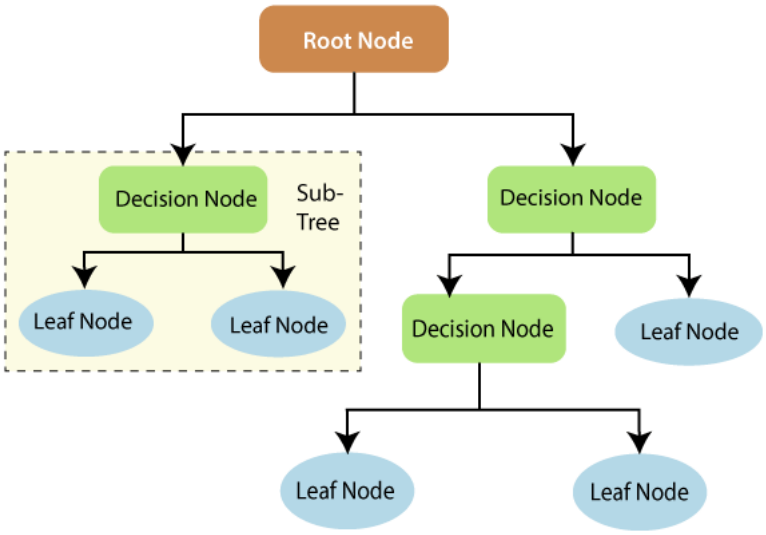
$$P(T_j|d_1, d_2,..., d_l) = \frac{\prod_{s=1}^{l} P(d_s|T_j)P(T_j)}{P(d_1, d_2,..., d_l)}, \qquad 1 \leq j \leq n \tag{6.3}$$

Z predpokladu nezávislosti javov potom dostaneme zo vzťahu 6.2 nasledujúci vzorec 6.3 a ten sa už nazýva Naivný Bayesov klasifikátor.

Naivný Bayesov klasifikátor ako sme si ho zadefinovali vyššie môže fungovať iba v teoretickej rovine, no pre jeho aplikáciu v reálnom svete je nutné získať odhady pravdepodobností zo vstupných trénovacích dát, keďže ich v praxi nemáme. Tieto odhady následne slúžia ako základ pre klasifikáciu nových vstupov. Tento klasifikátor sa často používa na spamovú filtráciu v e-mailovej komunikácii, klasifikáciu článkov a podobne.

6.2 Rozhodovacie stromy

Architektúra rozhodovacích stromov je tvorená hierarchickou štruktúrou pravidiel reprezentovaných pomocou uzlov.



Obr. 6.1: Rozhodovací strom [31].

Na ilustrácii 6.1 je znázornená štruktúra rozhodovacieho stromu. Každý rozhodovací strom sa skladá z troch typov uzlov: koreňového uzla, vnútorných uzlov

a nakoniec listov. Počiatočným uzlom každého rozhodovacieho stromu je koreňový uzol, po ktorom nasledujú vnútorné uzly. Pre úlohu klasifikácie textu, vnútorné uzly reprezentujú jednotlive charakteristiky respektíve tokeny dokumentu. Finálne rozhodnutie musí prejsť cez všetky tieto uzly až k listom. Listy sú uzly v strome, ktoré nemajú potomkov, reprezentujú jednotlivé klasifikačné triedy do ktorých chceme vstupné data priradiť. Vetvy vedúce z interných uzlov označujú rozhodovacie pravidlo, podľa ktorého sme sa v danom uzle rozhodli. Postupným prechodom rozhodovacím stromom od koreňového uzla po listy, pre každý dokument v súbore dát docielime jeho klasifikáciu podľa listového uzlu. Listový uzol priradí triedy na základe naučených hodnôt na trénovacej množine.

Najčastejšia verzia rozhodovacích stromov sa nazýva binárna. Pre tieto stromy platí, že každý vnútorný uzol má práve dvoch nasledovníkov (potomkov), práve z tejto vlastnosti vyplýva aj názov. Táto množina rozhodovácich stromov sa rozhoduje na základe prítomnosti alebo naopak absencie jednotlivých znakov v texte, ktorého triedu chceme určiť. Na obrázku 6.1 môžeme vidieť príklad takéhoto stromu. Rozhodovacie stromy však nemusia byť len binárne existujú aj terciárne a viac árne verzie rozhodovacích stromov pre klasifikačné úlohy. Obecná verzia rozhodovacieho stromu dokáže pracovať aj s reálnymi hodnotami a v tomto prípade hovoríme o regresných stromoch.

Rozhodovacie stromy používajú kľúčové charakteristiky pre klasifikáciu vzorov do správnej triedy. V našom prípade detekcie fake-news to popíšeme na výbere dokumentu repsektíve textu. V procese klasifikácie je kritické v každom kroku vybrať ten atribút, ktorý najlepšie rozdelí množinu textov v danom uzle. To znamená, že je dôležité rozhodnúť, ktorý atribút sa použije v koreňovom uzle, ktorý v interných uzloch. Tak aby sme docielili čo najlepšiu klasifikáciu.

Giniho index GI a Entropia H sú najbežnejšie používané kritéria na výber správneho atribútu pre rozdelenie rozhodovacieho stromu v danom uzle.

Giniho index je určený vzťahom

$$GI = 1 - \sum_{i=1}^{n} p_i^2, \tag{6.4}$$

Vzorec pre výpočet Giniho indexu GI v rozhodovacích stromoch používa pravdepodobnosti tried p_i, ktoré určujú podiel príkladov priradených určitej triede v danom uzle k celkového počtu všetkých príkladov v tomto uzle. Po rozdelení uzla na dcérske uzly sa Giniho index vypočíta pre každý nový uzol samostatne.

Giniho index pre vnútorný uzol sa vypočíta ako súčet hodnôt Giniho indexu pre každý z dvoch dcérskych uzlov, vynásobený podielom počtu pozorovaní v danom dcérskom uzle a celkového počtu pozorovaní v materskom uzle. Vzorec 6.5 zjednodušene vyjadruje, ako sa počíta celková hodnota Giniho indexu pre daný vnútorný uzol po jeho rozdelení na u nových uzlov:

$$GI^* = \sum_{i=1}^{u} \frac{N_i}{N_t} GI(i), \tag{6.5}$$

Vzorec pre celkovú hodnotu Giniho indexu v danom vnútornom uzle zahrňuje počet dcérskych uzlov u, počet pozorovaní N_i v každom z týchto uzlov a celkový počet pozorovaní N_t v materskom uzle.

Druhým najbežnejším rozhodovacím kritériom, ktoré sa používa na výber atribútu štiepenia uzlu je Entropia:

$$H = -\sum_{i=1}^{n} p_i \log_2 p_i, \tag{6.6}$$

Entropia je definovaná pomocou vzťahu 6.6, kde p_i predstavuje podiel pozorovaní priradených triede i v danom uzle vzhľadom na celkový počet pozorovaní v tomto uzle.

Vzorec pre celkovú hodnotu Entropie pre daný uzol môžeme vyjadriť ako súčet entropie pre každý dcérsky uzol násobený podielom počtu pozorovaní v danom dcérskom uzle a celkového počtu pozorovaní v materskom uzle. Inými slovami, suma entropie pre každý dcérsky uzol je násobená váhou, ktorá udáva podiel počtu pozorovaní v danom dcérskom uzle vzhľadom k celkovému počtu pozorovaní v materskom uzle.

$$H^* = \sum_{i=1}^{u} \frac{N_i}{N_t} H(i), \tag{6.7}$$

Symbol u znovu značí počet uzlov, podobne N_i značí počet pozorovaní v danom dcérskom uzle a symbol N_t značí celkový počet pozorovaní v materskom uzle. $H(i)$ reprezentuje hodnotu Entropie pre daný dcérsky uzol.

GI vs Entropia Pri porovnaní využitia GI vs Entropia v rozhodovacích stromoch ako kritéria delenia uzlov sa vo väčšine aplikuje viac práve Gini index. Táto skutočnosť je spôsobená štýlom výpočtu Entropie, ktorá je založená na logaritmickej funkcii. Výpočet tejto funkcie je náročnejší a preto sa v praxi viac používa Gini index [32]. Pre urýchlenie výpočtov pomocou rozhodovacích stromov.

6.2.1 Random Tree

Pri tomto algoritme je rozhodovaci strom, ktorý reprezentuje výsledný klasifikátor, vytvorený spôsobom náhodného výberu. Princíp spočíva v tom, že z množiny všetkých možných rozhodovacích stromov s k parametrami, ktoré riešia danú úlohu, sa náhodne vyberie jeden. Výber stromu sa realizuje tak, že každý strom v množine má rovnakú pravdepodobnosť byť zvolený ako výsledný klasifikátor. Inými slovami, výber rozhodovacieho stromu je náhodný a všetky stromy majú rovnakú šancu byť vybrané.

Zvyčajne spojenie väčšieho množstva náhodných stromov s nezávislosťou jednotlivých klasifikátorov zaručuje vytvorenie modelu, ktorý správne rieši danú klasifikačnú úlohu.

6.2.2 Random Forest

Algoritmus strojového učenia nazývaný Random Forest je založený na teórii rozhodovacích stromov. Algoritmus bol vytvorený ako riešenie problému pretrénovania, ktorý sa často vyskytuje pri použití rozhodovacích stromov v strojovom učení. Tento problém je spôsobený tým, že učiace sa algoritmy sa učia vzťahy medzi príkladmi a cieľovou triedou na základe trénovacej množiny dát, a to môže

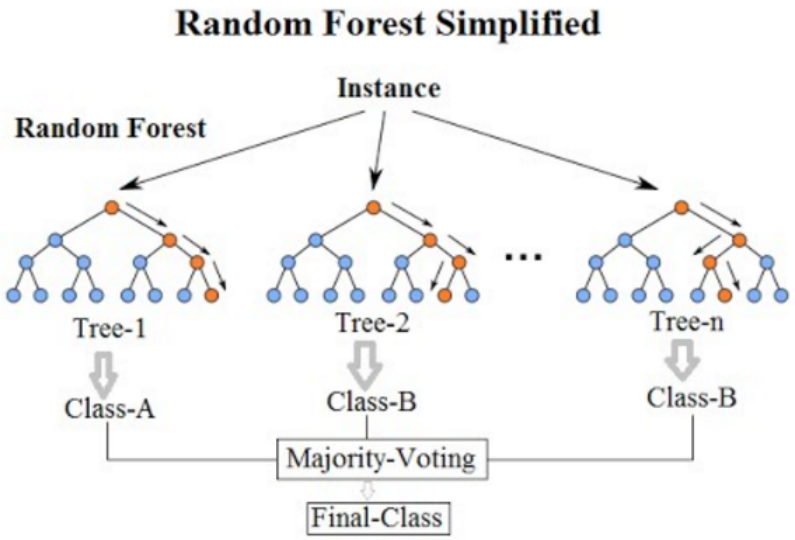
viesť k nadmernej prispôsobivosti a zníženiu výkonnosti pri testovaní na nových dátach [33].

Random Forest rieši tento problém vytvorením viacerých stromov, ktoré sú potom kombinované pomocou štatistického modusu. Takto vytvorený klasifikátor je robustnejší a spoľahlivejší, taktiež má lepšie schopnosti generalizácie a výkonu pri testovaní na nových dátach.

Vo formálne definícii sa Random Forest skladá z T_1,...,T_N množiny rozhodovacích stromov. Následne pre každý rozhodovací strom T_i patriaci do náhodneho lesa je vytvorená samostatná trénovacia množina, ktorá je vytvorená prevzorkovaním z kompletnej pôvodnej množiny dát. Ide o výbery s náhodným opakovaním príkladov, pričom je definovaná ich veľkosť n. Výstup klasifikácie pomocou tohto algoritmu strojového učenie môžeme vyjadriť vzťahom [34].

$$C_r f = modus\{C_i(x)\}_i^N, \tag{6.8}$$

kde N označuje počeť rozhodovacích stromov a $C_i(x)$ je predikcia i-teho stromu pre vstupné dáta x [35].



Obr. 6.2: Schéma algoritmu Náhodný les (Random Forest) [36].

Na obrázku 6.2 je ilustrácia ensemble systému s názvom náhodny les. Klasifikácia pomocou tohto modelu strojového učenia sa uskutoční kombináciou výsledkou náhodných stromov, ktoré sú súčasťou tohto ensebmle systému. Na nami ilustrovanom obrázku teda kombináciou výsledkov troch rozhodovacich stromov. Jednou z možností ako vybrať finálny výstup je hlasovanie väčšiny, pri klasifikácii alebo priemer hodnôt dielčich klasifikátorov pri regresných úlohach.

Náhodne lesy sa tešia značnej popularite medzi ensemble systémami. Toto rozšírenie algoritmu rozhovacieho stromu sa vyznačuje jednoduchou implementáciou. Ale na druhú stranu veľmi dobre sa vie vysporiadať s problémom preučenia.

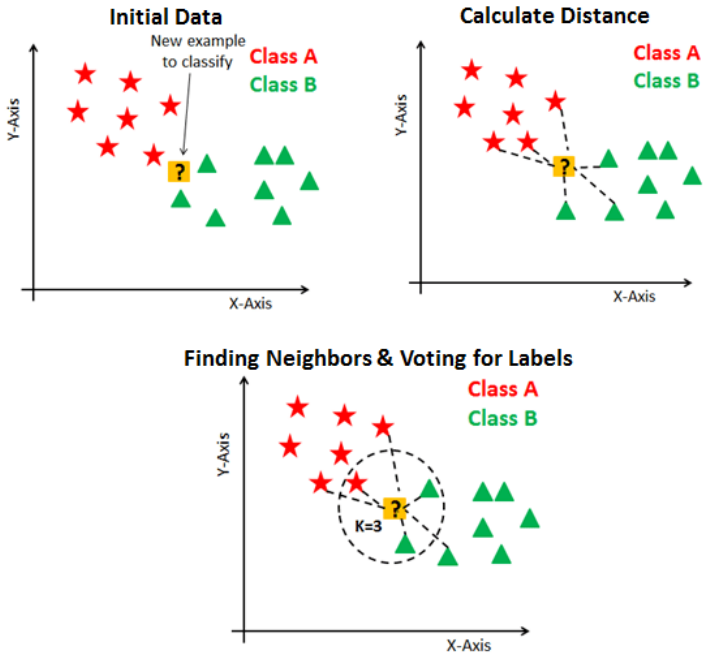
6.3 k-NN klasifikátor

Algoritmus k-NN, čo znamená k-najbližších susedov, pracuje s predpokladom, že každý vzor z trénovacieho dátového súboru reprezentuje bod v tomto n-dimenzionálnom priestore. Tieto vzory sú vyjadrené ako vektory v tomto priestore a každý z trénovacích vzorov je priradený k určitej klasifikačnej triede.

Pri trénovaní klasifikátora k-NN sa ukladajú vektory a prislúchajúce triedy trénovacích príkladov. Pri klasifikácii sa vyberie k najbližších susedov testovaného príkladu zvyčajne pomocou Euklidovskej vzdialenosti a rozhodnutie o triede testovaného príkladu sa robí na základe najčastejšej triedy medzi jeho najbližšími susedmi uloženými v procese trénovania [37].

Algoritmus k-NN pracuje s vstupným príkladom x, ktorý sa reprezentuje ako n-tica x_i, pričom i značí poradie vstupného vzoru. Trénovacia množina obsahuje dvojice (x_i, w_i), kde w_i označuje triedu, ku ktorej daný príklad patrí.

Klasifikátor k-NN zaradí vstupný príklad x do rovnakej triedy ako jeho najbližší príklad v trénovacej množine na základe funkcie $f(x) = w_i$. Najbližší príklad sa určí pomocou Euklidovskej vzdialenosti $min_{j\in 1,\dots,m}(x - x_j)$. Parameter k určuje, koľko najbližších susedov sa zapojí do klasifikácie vstupného príkladu x.



Obr. 6.3: Algoritmus k-NN [38].

k-NN bol pôvodne navrhnutý na riešenie klasifikačných problémov, ale nenáročnou modifikáciou ho možno upraviť pre výpočet spojitej cieľovej funkcie. Táto úprava spočíva v tom, že namiesto najčastejšie sa vyskytujúcej hodnoty sa vypočíta priemerná hodnota k-najbližších susedov z trénovacej množiny. Týmto spôsobom je algoritmus schopný riešiť regresné problémy.

Klasifikátor k-NN je implementačne pomerne nenáročný a poskytuje napriek tomu značne vysokú presnosť, najmä pre klasifikáciu s malým počtom tried. Existujú rôzne úpravy algoritmu k-NN, ktoré umožňujú jeho prispôsobenie pre rôzne potreby a požiadavky. [39].

6.4 Ensemble metódy strojového učenia

V dnešnej dobe sa strojové učenie výrazne rozšírilo a používa sa na riešenie rôznych úloh. Napriek jeho úspechu však podrobnejší výskum ukázal rôzne problémy, ktoré súvisia predovšetkým s nedostatkom dát, veľkou variabilitou dát a takzvaným preučením metód.

Ensemble stratégia je jednou z metód, ktoré sa využívajú na riešenie vyššie uvedených problémov a dosahuje pomerne dobré výsledky. Táto metóda kombinuje výstupy viacerých klasifikátorov namiesto použitia jedného samostatného klasifikátora. Tento prístup môže viesť k protichodným výsledkom, ktoré zvyšujú presnosť a robustnosť predikcie.

Mnohé teoretické a praktické dôvody hovoria v prospech použitia ensemble systémov [40]. často spomínanými dôvodmi pre použitie tejto metódy sú malá respektíve veľká dátova sada, performačné rozdiely algoritmov a v neposlednom rade zložitosť problému.

Existuje niekoľko rôznych techník ensemble learningu. Dve základné metódy ensemble strojového učenia sú Bagging a Boosting. Po naučení jednotlivých klasifikátorov v ensemble systémoch je dôležité zvoliť najvhodnejší spôsob spojenia ich výsledkov.

6.4.1 Bagging

Bagging, známy aj ako bootstrap aggregation, je jednou z najstarších metód ensemble systémov, ktoré sa používajú na zlepšenie presnosti algoritmov strojového učenia pre klasifikačné a regresné úlohy. Využíva náhodný výber s opakovaním z pôvodného trénovacieho súboru dát na vytvorenie bootstrap podmnožín, ktoré sa používajú na trénovanie rôznych klasifikátorov. Výstupom je hlasovanie väčšiny z jednotlivých klasifikátorov, pričom konečné rozhodnutie je určené triedou s najväčším počtom hlasov od klasifikátorov tvoriacich systém. Bagging je veľmi jednoduchý na implementáciu a je účinný najmä v prípade obmedzeného množstva dát [41].

6.4.2 Boosting

Boosting je algoritmus, ktorý sa používa na výber modelov pre ensemble systémy. Tento algoritmus umožňuje transformovať slabý klasifikátor, ktorého úspešnosť je iba o niečo lepšia ako náhodného, na silný klasifikátor, ktorý poskytuje správne predikcie pre všetky príklady [42].

Pri použití boostingu sa vytvára súbor klasifikátorov, ktoré sú natrénované na prevzorkovanom súbore dát s použitím hlasovania väčšiny, rovnako ako pri metóde bagging. Avšak, boosting priniesol vylepšenie v spôsobe prevzorkovania, kde sa každému klasifikátoru poskytne čo najviac informatívne trénovacie dáta. Toto sa dosahuje napríklad tým, že sa zvýši váha pre príklady, ktoré boli zle klasifikované v predchádzajúcom kroku trénovania, aby sa k nim klasifikátor snažil priradiť správnu triedu. Takýmto spôsobom sa zvyšuje pozornosť k zložitým prípadom a zlepšuje sa výkonnosť klasifikátora.

7. Hlboké neurónové siete

Umelé neurónové siete tiež skrátene nazývane neuronové siete sú v súčasnosti považované odbornou verejnosťou za jeden z najlepších algoritmov strojového učenia. V dnešnej dobe žnú mnohé úspechy v rôznorodých úlohách [43]. Prinášajú výsledky s veľmi vysokou presnosťou a majú pomerne širokú škálu použitia. Limitované sú iba množinou zdrojových dát, ktoré sú potrebné pre ich správne naučenie. Možnými aplikáciami neuronových sieti sú úlohy počítačového videnia či spracovanie prirodzeného jazyka [44].

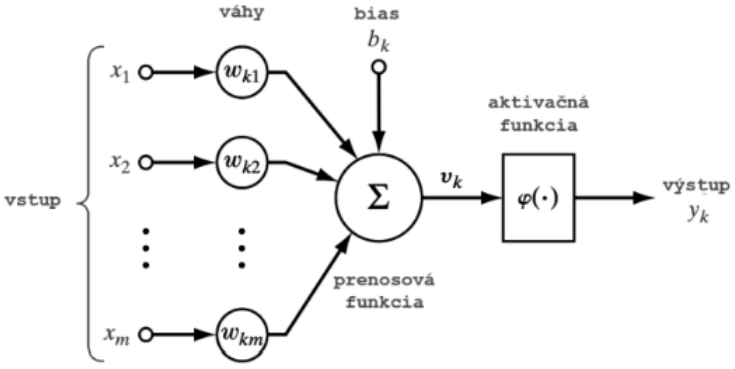
V závislosti od použitej architektúry ich niekedy možno označiť aj ako hlboké strojové učenie, ktoré je s narastajúcim výpočtovým výkonom čoraz populárnejšie a komplexnejšie. Medzi najznámejšie neurónové siete, ktoré sú známe aj širokej verejnosti patria modely od spoločnosti OpenAI, ktorými sú GPT.

Na rovnakom princípe ako mnohé iné algoritmy strojového učenia aj neurónové siete fungujú tak, že najskôr sa vytvorí model (architektúra), ktorý sa v ďalšom kroku natrénuje na čo najrozsiahlejšom množstve anotovaných dát. Na základe vzťahov v týchto známych trénovacích dátach sa model naučí predpovedať výsledok pre nové neznáme príklady z rovnakej domény [43].

V roku 2010 sa učenie reprezentácie a metódy strojového učenia v štýle hlbokej neurónovej siete rozšírili do spracovania prirodzeného jazyka, čiastočne vďaka množstvu výsledkov, ktoré ukazujú, že takéto techniky môžu dosiahnuť najmodernejšie a najlepšie výsledky v mnohých úlohách prirodzeného jazyka ako napríklad pri modelovaní jazyka, analýze a mnohých ďalších.

7.1 Architektúra neurónových sietí

Neurónové siete sú jedným z matematických modelov strojového učenia, ktoré používajú interné prenosové funkcie na transformáciu vstupných vektorov na výstupné. Ich architektúra a fungovanie sa inšpiruje fungovaním ľudského mozgu [44], preto aj mnohé termíny v tejto oblasti umelej inteligencie sú inšpirované ich anatomickými ekvivalentami.



Obr. 7.1: Vizualizácia umelého neurónu [44].

Na ilustrácii 7.1 je vyobrazená základná výpočtová jednotka neuronových sieti, ktorá sa nazýva umelý neurón. Podobne ako neúronové siete aj ich vý-

počtová jednotka je inšpirovaná biologickými nervovými bunkami. Umelý neurón predstavuje hrubý model biologického neurónu a snaží sa aproximovať jeho chovanie. V literatúre je neurón definovaný pomocou váh, bias faktora a aktivačnej funkcie, ktoré sa používajú na transformáciu vstupných dát.

V nasledujúcom odstavci si popíšeme zľava do prava ilustráciu na obrázku 7.1, ktorá reprezentuje architektúru umelého neurónu. Vstupné hodnoty sú reprezentované premennými $x_1, ..., x_n$, ktoré vstupujú do neurónu spolu s ich váhami $w_1, ..., w_n$ pre výpočet. No ilustrácii sú vyznačené šípkou, ktorá smeruje do vnútra umelého neurónu.

V umelom neuróne, atribút x_0 a jemu prislúchajúca váha w_0 označujú špeciálny atribút nazývaný bias, ktorého úlohou je ovplyvniť výstup aktivačnej funkcie neurónu. Podobne ako v biologických neurónoch, aj v umelých neurónoch existuje vnútorný potenciál označený ako z, ktorý sa skladá z váh a vstupov s dodatočným zohľadnením biasu. Vzťah 7.1 je matematickou interpretáciou výpočtu potenciálu neurónu, ako sme ho popísali.

$$z = w_0 x_0 + \sum_{i=1}^{n} w_i x_i \tag{7.1}$$

Výstup umelého neurónu sa získava kombináciou jeho vnútorného potenciálu a aktivačnej funkcie g, čo je vyjadrené pomocou nasledujúceho vzorca 7.2.

$$y = g(z) \tag{7.2}$$

Aktivačná funkcia neurónu sa používa na prahovanie vnútorného potenciálu. Pôvodne sa využívala ostrá nelinearita, no neskôr ju nahradili sigmoidálne funkcie ako $\sigma(x)$ alebo $\tanh(x)$, ktoré dokážu lepšie reprezentovať komplexnejšie funkcie a sú jednoducho diferencovateľné [45].

Aktivačné funkcie		
Názov	Obor hodnôt	Funkcia
Ostrá nelinearita	$\langle 0,\ \infty \rangle$	$g(x) \begin{cases} 1 & x > 0.5 \\ 0 & x \leq 0.5 \end{cases}$
ReLu	$\langle 0,\ \inf)$	$g(x) = max(0,x)$
Logistická sigmoida	$(0,\ 1)$	$g(x) = \frac{1}{(1+e^{-}x)}$
Hyperbolický tangens	$(-1,\ 1)$	$g(x) = \frac{e^{x}-e^{-}x}{e^{x}+e^{-}x}$

Tabuľka 7.1: Základné aktivačné funkcie neurónových sieti.

7.2 Perceptron

Jeden z najjednoduchších modelov strojového učenia je Perceptron, ktorý tvorí jeden umelý neurón s aktivačnou funkciou typu $\sigma(x)$ alebo prípadne ostrá nelinearita. Tento model dokáže reprezentovať jednoduchý binárny klasifikátor [46].

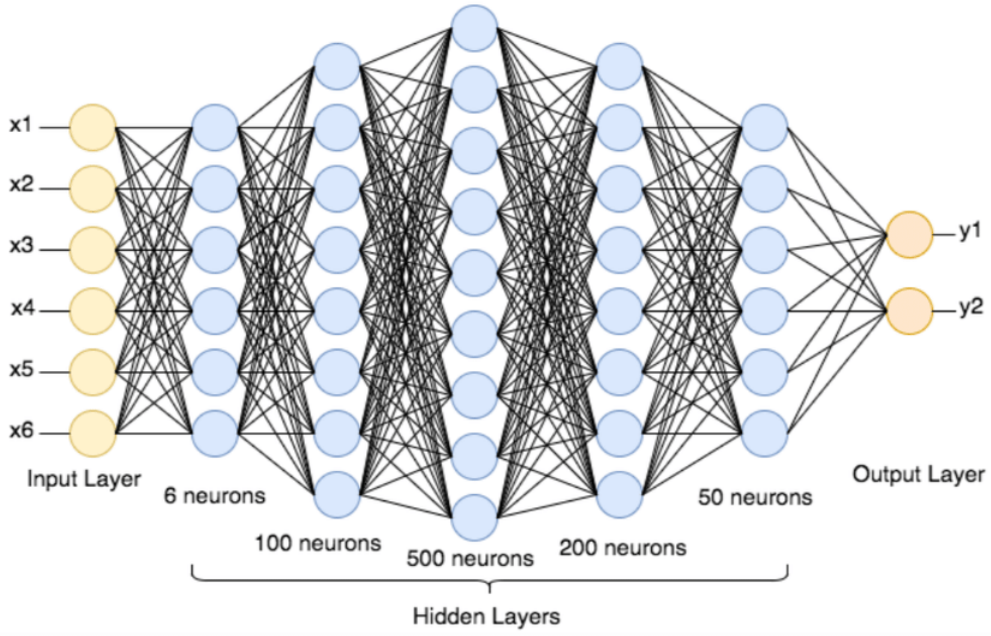
7.3 Neurónová sieť

Perceptron má nevýhodu, že je schopný správne rozdeliť a klasifikovať iba dáta, ktoré sú lineárne separovateľné. To znamená, že v prípade úloh z reálneho sveta, kde sú dáta zvyčajne zložitejšie a ne-lineárne, nie je dostatočným riešením. Dokonca aj jednoduchá funkcia ako XOR je pre perceptron problémom, pretože nie je lineárne separovateľná. Aproximácia už tejto jednoduchej funkcie pomocou perceptronu je nemožná [44].

Použitie väčšej množiny neurónov a ich zapojenie do komplexnejšieho systému umožňuje prekonať obmedzenia Perceptronu. Spôsobom spájania neurónov do usporiadanej štruktúry sa vytvára umelá neurónová sieť.

7.3.1 Topológia neurónovej siete

Neurónová sieť pozostáva z neurónov usporiadaných vo vrstvách. Základná architektúra siete sa skladá z vstupnej a výstupnej vrstvy, pričom komplexnejšie neurónové siete majú aj skryté vrstvy, ktoré sú uložené medzi vstupnú a výstupnú vrstvu. Skryté vrstvy predstavujú časti siete, kde sa spracúvajú a transformujú vstupné dáta. Hlavne umožnujú neurónovým sieťam spracovať zložitejšie úlohy, ktoré by nebolo možné riešiť len s vstupnou a výstupnou vrstvou.



Obr. 7.2: Dopredná neurónová sieť [47].

Na ilustrácii 7.2 je vyobrazená topológia respektíve architektúra takejto siete. Neurónová sieť je tvorená vrstvami, ktoré pozostávajú s neurónov. Pokiaľ informácie prechádzajú cez neuróny jednotlivých vrstiev iba v jednom smere, jedná sa o doprednú neurónovú sieť. Dopredné šírenie signálov medzi neurónmi jednotlivých vrstiev nie je nevyhnutnou podmienkou neurónových sietí. Rekurentné siete sú schopné navyše obsahovať spätné prepojenia medzi neurónmi a vrstvami, čo umožňuje informáciám cirkulovať a meniť stav siete v čase.

Pojmy ako architektúra, topológia alebo organizačná dynamika siete sa v literatúre používajú na rozdelenie neurónových sietí na rekurentné a dopredné siete.

Avšak, v oblasti umelej inteligencie zaoberajúcej sa neurónovými sieťami existuje množstvo rôznych prístupov k terminológií, pretože ide o mladú a dynamickú oblasť. Terminológia môže byť rôznorodá a môže sa líšiť v závislosti od zdrojov a literatúry, ktoré sa používajú.

7.3.2 Učenie neurónových sietí

Podobne ako ich biologický vzor, aj umelé neurónové siete majú schopnosť učiť sa, čo je ich kľúčovou vlastnosťou. V kontexte strojového učenia to znamená, že sieť môže byť natrénovaná na riešenie špecifického problému [44].

Neurónová sieť sa učí iteratívne na trénovacom súbore dát a jednou z najznámejších metód učenia je algoritmus spätného šírenia chyby (Backpropagation). Po každom prechode súboru dát sa vypočíta gradient chybovej funkcie, ktorý sa určí na základe spätného prechodu sieťou. Gradient sa použije na úpravu váh neurónov tak, aby sa minimalizovala hodnota chybovej funkcie. Chybová funkcia, tiež označovaná ako loss sa vypočíta podľa vzťahu 7.3 a vyjadruje rozdiel medzi predikovaným a skutočným výstupom neurónovej siete.

$$J(W) = \sum_{i=1}^{n} l(x^{(i)},y^{(i)},W) + \lambda R(W) \tag{7.3}$$

Funkcia $l(x^{(i)},y^{(i)},W)$ určuje chybu pre jeden príklad v trénovacej množine. Potom $J(X,Y,W)$ vyjadruje celkovú chybu pre všetky trénovacie príklady. V druhej časti vzorca je implementovaná regularizácia, ktorá slúži na doplnkovú penalizáciu pre zlepšenie výkonnosti modelu.

Pri trénovaní neurónových sietí hrajú významnú úlohu rôzne hyperparametre, ktoré ovplyvňujú proces učenia [48]. Okrem zmien v štruktúre siete, ako je napríklad inicializácia váh, počet skrytých vrstiev a ich veľkosti, voľba aktivačných a chybových funkcií, existujú aj ďalšie hyperparametre, ktoré majú vplyv na trénovanie neurónovej siete:

- Počet epoch určuje, koľkokrát sa sieť naučí na celú trénovaciu množinu.
- Veľkosť skupiny (angl. batch size) je počet prvkov z trénovacej sady, ktoré sa používajú na úpravu váhy neurónov.
- Miera učenia (angl. learning rate) znamená rýchlosť úpravy váh neurónov po každom kroku trénovania.

V ďalšom odseku popíšeme algoritmus spätného šírenia chyby, ktorý sa využíva pri učení neurónových sietí. Popis pre jednoduchosť uvedieme pre jednu epochu, čo znamená jeden prechod trénovacou sadou. Pri skutočnom tréningu neurónovej siete by bolo nutné niekoľkokrát tento proces zopakovať.

Začiatok učenia neurónových sietí zahŕňa ich počiatočnú inicializáciu. Ak by sme však inicializovali váhy neurónov rovnakými hodnotami, zostali by počas učenia nezmenené. Preto sa používa náhodná inicializácia váh z intervalu $(-e, ; e)$, pričom e je malé číslo z intervalu (0, 1). Existujú aj algoritmy na inteligentnú inicializáciu váh, ktoré vedú k lepšiemu učeniu neurónovej siete.

Po inicializácii neurónovej siete sa trénovacie dáta prechádzajú postupne po jednom príklade x_i. Pri tomto prechode sa vypočíta dopredný prechod siete. Na

základe výstupu siete sa potom vypočíta chyba δ, ktorá sa iteratívne počíta späť pre ďalšie skryté vrstvy.

Po každom trénovacom príklade sa vypočítajú gradienty na základe chyby pre každú vrstvu neurónovej siete. Tieto gradienty sú potom použité na úpravu váh siete v rámci metódy učenia, ktorú sme zvolili. Ak sa použije postupné akumulovanie gradientov, úprava váh sa uskutoční po dokončení prechodu cez celý trénovací súbor dát. Ak sa použije online učenie, váhy siete sa upravujú okamžite po každom trénovacom príklade.

7.3.3 Transfer learning

Transfer learning je metóda strojového učenia, ktorá spočíva v použití predtrénovanej neurónovej siete na riešenie novej úlohy. Aby bola táto sieť úspešná a dokázala dobre reprezentovať príznaky, je potrebné, aby bola trénovaná na veľkom súbore vstupných dát. Pod pojmom „veľký súbor dát“ sa rozumie niekoľko stoviek gigabajtov až terabajtov dát.

Učenie neurónovej siete na takomto rozsiahlom súbore dát môže odhadovo trvať niekoľko týždňov, aj farme s grafickými kartami. Avšak jeho následné použitie už je jednoduché a časovo nenáročne, vzhľadom k tomu, že takto natrénované modely sú verejne dostupné.

Pri konvolučných a rekurentných sieťach sa najčastejšie používajú dva prístupy pri transfer learningu. Prvým prístupom je fine-tuning, ktorý spočíva v jemnej úprave už natrénovanej siete pre nový problém. Druhým prístupom je použitie siete ako extraktoru príznakov.

Sieť ako extraktor príznakov

Jeden z možných prístupov ako použiť predtrénovaný model neurónovej siete spočíva v tom, že sa ako vstup pre ďalšiu metódu strojového učenia použije výstup niektorej z vrstiev predtrénovanej siete. Tento vektor príznakov, ktorý je extrahovaný z tejto siete sa tiež nazýva „descriptor“.

Je bežnou praxou využiť výstup z určitej vrstvy predtrénovanej neurónovej siete ako vektor príznakov pre iný model strojového učenia. Tento prístup sa často uplatňuje v úlohách ako klasifikácia obrázkov, pri ktorej môže posledná vrstva konvolučnej siete poskytnúť vstup pre algoritmus k-NN.

Fine-tuning

Druhou veľmi častou metódou transfer learningu je Fine-tuning. Základom tejto techniky je použitie predtrénovaného modelu neurónovej siete alebo iba váh vrstiev na riešenie nového problému. Počas fine-tuningu stačí zmeniť iba niektoré vrstvy v tejto sieti, obvykle poslednú alebo posledné dve vrstvy. Váhy neurónovej siete, ktoré sa nepotrebujú meniť, sa zamrazia a to urýchľuje učenie siete na novom súbore dát.

Konkrétnym príkladom použitia techniky fine-tuning je situácia, kedy sa využije konvolučná neurónová sieť natrénovaná na datasete Imagenet. Tento model má v svojich váhach pre jednotlivé vrstvy zakódované poznatky, ktoré sú schopné správne detekovať tvary, farby prípadne hrany na obrázku. Tieto už naučené

poznatky môžu byť následne využité pri riešení nových problémov z oblasti rozpoznania obrazu akou je napríklad detekcia objektov. Všeobecné znalosti získané z predchádzajúcej trénovacej sady umožňujú rýchlejšie prispôsobenie sa novému problému.

7.4 Konvolučné neurónové siete

Aktuálne sú konvolučné neurónové siete (CNN) jednou z najpopulárnejších foriem neurónových sietí. Svoje najväčšie uplatnenie našli v oblasti počítačového videnia. Okrem toho sa však objavujú aj úspešné aplikácie v spracovaní prirodzeného jazyka a reči, na ktoré v tejto práci budeme chcieť nadviazať.

Primárna oblasť ich využitia nie je prekvapivá, vzhľadom k tomu že architektúra CNN je inšpirovaná fungovaním vizuálneho centra v mozgu cicavcov, čo znamenalo významný prínos v oblasti strojového učenia [49]. Lokálne prepojenia predstavujú základnú vlastnosť vizuálneho centra mozgu, ktorá bola prebraná do konvolučných neurónových sietí. Lokálne prepojenia umožňujú neurónom v skrytej vrstve sa pripojiť k oblasti neurónov v predchádzajúcej vrstve, a to cez takzvané receptive field. Umožňuje to týmto meurónovým sieťam lepšie zachytiť lokálne rysy [50]. Táto schopnosť, môže byť žiadúca pre niektoré úlohy spracovania prirodzeného jazyka [51], aj napriek tomu, že konvolučné siete boli pôvodne navrhnuté pre spracovanie vizuálneho obsahu. Druhou dôležitou vlastnosťou týchto sietí je n-dimenzionálny rozmer vrstiev. Napríklad pre vizuálne dáta majú tri dimenzie pre každý kanál spektra RGB. Poslednou ale nie menej dôležitou vlastnosťou je zdieľanie váh neurónov v konvolučnej vrstve v danej hĺbke, čo znamená, že všetky neuróny v tejto vrstve majú rovnaké váhy.

Konvolučné neurónové siete majú výhodu oproti klasickým neurónovým sieťam vďaka vlastnostiam, ktoré sme popísali v predchádzajúcich odstavcoch. Ak by sme sa rozhodli klasickou doprednou sieťou klasifikovať vizuálne dáta, museli by sme vytvoriť veľké množstvo neurónov pre každý pixel obrázka, s obrovským počtom prepojení na nasledujúcu vrstvu. Trénovanie siete s takouto architektúrou by bolo časovo náročne a sieť by mala tendenciu k preučeniu, práve z tohto dôvodu vznikli konvolučné siete, ktorých základnými vlastnosťami sa snažíme tieto problémy potláčať.

Tento typ siete má schopnosť naučiť sa zložité vzorce a vlastnosti z vstupných dát. V prvých vrstvách siete sa učia detekovať jednoduché prvky ako čiary a hrany. Postupne sa naučia rozpoznávať zložitejšie prvky, ako kruhy a viacfarebné prechody. V posledných vrstvách siete sa už učia rozpoznávať veľmi zložité tvary a objekty na vstupnom obrázku.

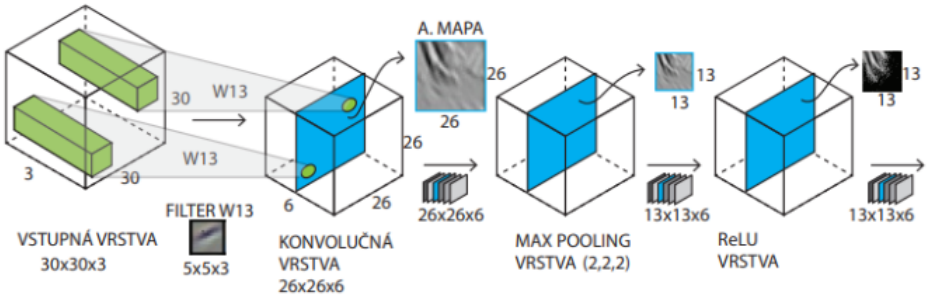
CNN sa skladajú z vstupnej vrstvy a blokov, ktoré zahŕňajú vrstvy konvolúcie, poolingu a aktivačnú funkciu ReLu [52]. Pred výstupnou vrstvou siete sa nachádza plne prepojená vrstva neurónov, ktorá vyžaduje zhustenie výstupu z konvolučných blokov do jednej dimenzie. Výstup CNN siete sa častokrát používa ako vstup (descriptor) pre iné metódy strojvého učenia v procese zvanom transfer learning.

7.4.1 Konvolučná vrstva

Táto vrstva siete vykonáva konvolučnú operáciu, ktorá spočíva v umiestnení filtru nad vstupnou maticou. Konvolučný filter je umiestnený tak, aby sa jeho

horný ľavý roh prekrýval s horným ľavým rohom vstupnej matice. Následne sa hodnoty z vstupnej matice postupne násobia s príslušnými hodnotami z konvolučného filtra a súčet týchto hodnôt sa umiestni na príslušné miesto vo výslednej matici. Filter sa potom posúva po celej vstupnej matici a tento proces sa opakuje, až kým nie sú pokryté všetky prvky vstupu [52].

Na ilustrácii 7.3, môžeme vidieť vizualizáciu fungovania konvolučnej vrstvy na obrazových dátach.

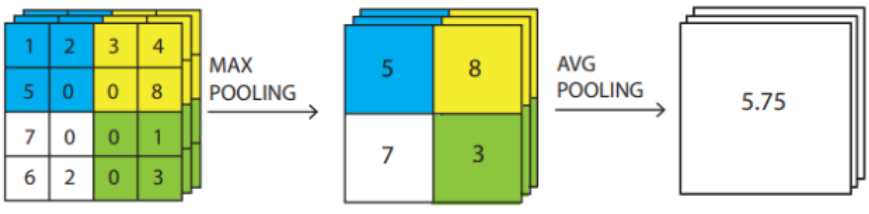


Obr. 7.3: Schématická vizualizácia konvolučnej siete[50].

7.4.2 Pooling vrstva

Pooling vrstva je kľúčovou súčasťou konvolučných sietí, ktorá slúži na zmenšenie priestoru vstupov z predchádzajúcich vrstiev. Tento proces znižuje počet parametrov v nasledujúcich vrstvách a zlepšuje rýchlosť učenia siete. Zároveň pomáha zlepšiť generalizáciu modelu [52].

Pooling vrstva má dva dôležité parametre: veľkosť okna/filtru a posun. Typické hodnoty sú 2 pre veľkosť filtra s rovnako veľkým posunom. Jednotlivé časti vstupnej mapy sa podrobia preddefinovanej operácii a pre každú časť sa vyberie reprezentatívna hodnota na základe zvoleného typu filtra. Maximálna hodnota z oblasti predstavuje MaxPooling vrstvu, zatiaľ čo priemerovanie hodnôt predstavuje AvgPooling.



Obr. 7.4: Operácia MaxPooling a AvgPooling pre data[50].

Pooling vrstva nepoužíva žiadne váhy a slúži iba na transformáciu vstupu na výstup, podobne ako aktivačná vrstva. Na ilustrácii 7.4 je znázornený priebeh tejto operácie.

7.4.3 ReLu vrstva

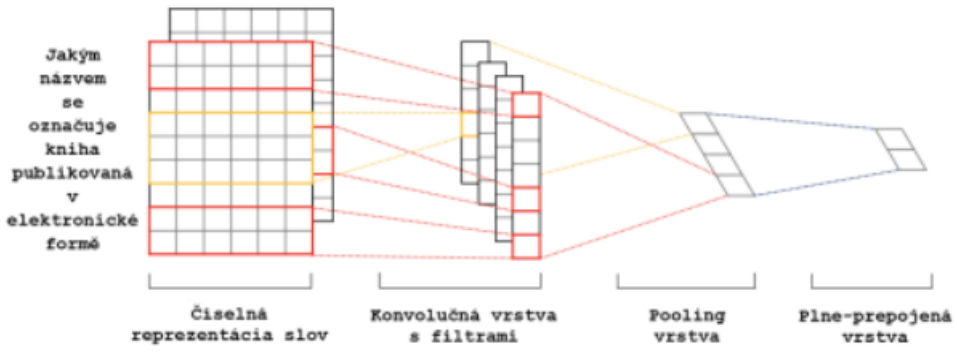
ReLU vrstva sa používa ako aktivačná vrstva v konvolučných sieťach a je oddelená do samostatnej vrstvy. Táto oddelená aktivačná vrstva umožňuje použiť aktivačnú funkciu na rôzne vrstvy, ako sú konvolučné vrstvy, pooling vrstvy alebo plne prepojené vrstvy. ReLU vrstva je preferovaná pred sigmoidálnymi funkciami kvôli jej rýchlemu výpočtu a schopnosti riešiť problém miznúceho gradientu. ReLU vrstva sa často používa hneď po konvolučnej alebo plne prepojenej vrstve [52].

7.4.4 Plne-prepojená vrstva

V plne prepojenej vrstve v CNN sa neuróny spoja so všetkými výstupmi z predchádzajúcej vrstvy. Váhy neurónov sa učia počas trénovania, aby dokázali identifikovať dôležité vzorce vo vstupných dátach. Takto sa vytvára hierarchia extrakcie príznakov, kde každá vrstva sa snaží identifikovať zložitejšie vzorce v dátach. Plne prepojená vrstva v CNN má za úlohu zlúčiť informácie z predchádzajúcich vrstiev a klasifikovať vstupné dáta do konkrétnych tried. Táto vrstva sa často používa v posledných vrstvách siete, kde sa na základe extrahovaných príznakov vykonáva klasifikácia vstupných príkladov. Teda cieľom plne prepojenej vrstvy je určiť skóre, ktoré indikuje priradenie vstupu do konkrétnej triedy.

7.4.5 Učenie siete

Trénovanie konvolučných sietí sa v zásade vo väčšine prípadov nelíši od trénovania dopredných sietí. Prevažná množina konvolučných sietí sa učí pomocou algoritmu spätného šírenia chyby, rovnako ako klasické dopredné siete.



Obr. 7.5: Vizualizácia CNN na textových dátach [50].

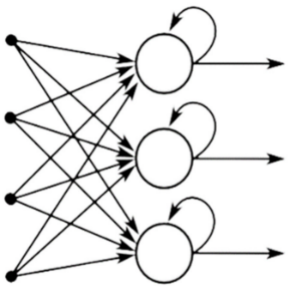
Využitie konvolučných neurónových sietí sa rozšírilo aj na spracovanie prirodzeného jazyka, pri ktorom sa text reprezentuje ako jednorozmerné pole slov. Na začiatku tohto procesu sa slová z textu transformujú na vektory čísel určitej dimenzie, na ktoré sa následne aplikujú konvolučné filtre prispôsobené pre jednorozmerné vstupy. Výsledné vektory z konvolučných vrstiev sa potom spracujú pomocou ReLu vrstvy a prejdú ďalšou vrstvou. Posledný krok v konvolučnom

bloku predstavuje aplikácia Pooling vrstvy, ktorá pre každý filter zvlášť vyberie maximálnu hodnotu z mapy rysov a túto hodnotu považuje za najdôležitejší rys daného filtra. Ak sa používa maximalizačný filter, extrahuje sa maximálna hodnota, pri použití AvgPooling filtra sa vyextrahuje priemerná hodnota [51].

Výstup z konvolučného bloku siete je ďalej predávaný do plne-prepojenej vrstvy, ktorej výstupom je distribúcia pravdepodobností priradenia do jednotlivých klasifikačných tried. Na ilustrácii 7.5 je vizualizácia CNN, ktorá je uvedená aj s príkladom vo forme textu.

7.5 Rekurentné neurónové siete

Rekurentná neurónová sieť je typ modelu, ktorý dokáže spracovávať sekvenčné dáta akejkoľvek dĺžky. Termínom sekvenčné dáta označujeme dáta obohatené o časový kontext. Príkladom týchto dát sú napríklad video záznamy alebo vývoj cien komodít na burze.



Obr. 7.6: Vizualizácia rekurentnej neurónovej siete.

Rekurentné neurónové siete (RNN) sa od klasických dopredných neurónových sietí líšia tým, že v RNN existujú väzby medzi jednotlivými jednotkami, ktoré vytvárajú cykly. Tieto cykly umožňujú sieti vziať do úvahy aj informácie z predchádzajúcich stavov. Schému architektúry RNN možno vidieť na vizualizácii 7.6.

Rekurentné neurónové siete majú výhodu v tom, že umožňujú prácu s vstupnými sekvenciami variabilnej dĺžky, čo je v konvenčných dopredných neurónových sieťach nemožné, ktoré môžu pracovať iba so vstupom fixnej dĺžky. Táto schopnosť je kľúčová pre RNN a vychádza z faktu, že váhy siete sú zdieľané pre všetky časti sekvencie, čo umožňuje flexibilnejšie spracovanie vstupných dát. Tento prístup poskytuje výhodu pre trénovacie a testovacie dáta s rozdielnou dĺžkou sekvencií.

Rekurentná neurónová sieť je schopná čiastočne si zapamätať predošlý kontext vstupných sekvencií x pomocou skrytého stavu a rekurentných spojení medzi neurónmi alebo vrstvami. To znamená, že neuróny v skrytej vrstve majú dva druhy spojení - okrem spojenia s ďalšími vrstvami, majú aj rekurzívne spojenia na seba. V každom kroku sa tieto spojenia používajú na aktualizáciu skrytého stavu, ktorý zodpovedá stavu neurónovej siete v danom čase t a zohľadňuje predošlý kontext.

Stav skrytej vrstvy h_t v RNN sa postupne aktualizuje a uchováva informácie o predchádzajúcich krokoch modelu, teda o kontexte, ktorý vstupné sekvencie prešli počas učenia. Vzniká ako výsledok rekurentného spracovania vstupov a kumuluje sa postupne do skrytého stavu pre ďalšie spracovanie.

Pre ilustráciu, ak máme rekurentnú neurónovú sieť s jedinou skrytou rekurentnou vrstvou, jej dopredný prechod sa skladá z troch krokov. V prvom kroku sa vstupná sekvencia prenesie do skrytého neurónu alebo vrstvy. V druhom kroku sa výstup z tejto skrytej vrstvy prenáša na výstup modelu. V treťom kroku sa výstup z tejto skrytej vrstvy prenáša do ďalšej skrytej vrstvy, kde sa použije ako vstup do ďalšieho kroku v rekurentnom spracovaní.

$$h_i = \sigma((W_{hh}h_{t-1}) + W_{xh}x_t + b_h) \tag{7.4}$$

Vzťah 7.4 vyjadruje stav h_t skrytej vrstvy v čase t, kde W_{xh} označuje spojenia respektíve váhy zo vstupu do skrytej vrstvy, W_{hy} zo skrytej vrstvy na výstup a W_{hh} rekuretne zo skrytej vrstvy. Potom výstup siete y v čase t sa vypočíta pomocou nasledujúceho vzorca:

$$y = \sigma(W_{hy}h_t) \tag{7.5}$$

Rekurentné neurónové siete priniesli množstvo výhod ale ich klasická verzia sa potýka so závažným problémom, ktorým je miznúci a explodujúci gradient. V angličtine známy pod pojmom „vanishing and exploding gradient problem“. Tento problém je spôsobený princípom učenia a architektúrou RNN [53]. V procese učenia modelu sa pri každom spätnom prechode neurónovou sieťou gradient chybovej funkcie vynásobi biasom. Pri učení modelu založeného na architektúre rekurentných neurónových sietí sa počas spätného šírenia chyby gradient chybovej funkcie násobí biasom. Tento bias je zvyčajne menší alebo väčší ako 1 pre väčšinu relevantných príkladov. Po určitom čase trénovania klasických RNN na súbore dát s dlhodobými závislosťami môže dôjsť k problému miznúceho alebo explodujúceho gradientu chybovej funkcie.

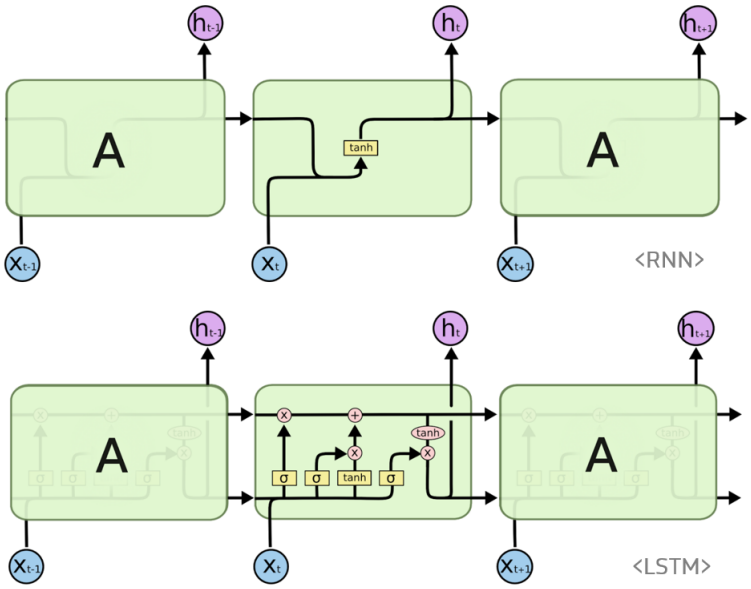
Klasické rekurentné neurónové siete sa v úlohách s dátami, ktoré majú dlhodobé závislosti, ukázali ako neefektívne kvôli problému miznúceho gradientu. Preto boli vyvinuté vylepšenia klasických rekurentných neurónových sieti, ktoré tento problém eliminujú, napríklad Long Short-Term Memory [53].

7.5.1 LSTM

Modifikácia architektúry rekurentnej neurónovej siete, známa ako Long Short-Term Memory (LSTM), rieši problémy s miznúcim a explodujúcim gradientom. Ktorý sa vyskytujú pri učení klasických rekurentých sieti, vďaka čomu sa dokážu efektívne učiť aj na dátach s dlhodobými závislosťami.

Vyriešenie problému miznúceho a explodujúceho gradientu v klasických RNN spočívalo v úprave architektúry samotnej výpočtovej jednotky tejto siete [53]. Pamäťová bunka v LSTM ukladá stav, ktorý sa modifikuje pomocou troch brán - vstupnej, pamäťovej a výstupnej. Každá brána je reprezentovaná sigmoidnou vrstvou, ktorá určuje, do akej miery má ovplyvniť súčasný stav pamäťovej bunky. Táto bunka je súčasťou reťazovej štruktúry rekurentnej neurónovej siete a je znázornená na obrázku 7.7.

V súvislosti s vývojom komplexnejších LSTM sietí pre spracovanie prirodzeného jazyka bolo publikovaných množstvo štúdií, ktoré sa zameriavali na použitie nových učiacich algoritmov a pokročilých architektúr, najmä na metódu Long Short Term Memory (LSTM), ktorú navrhli Hochtreiter a Schimhuber [53]. V svojej štúdii navrhli vytvoriť ako už sme spomenuli dodatočné pamäťové brány,



Obr. 7.7: Porovannie RNN vs. LSTM [54].

ktoré by boli zodpovedné za úpravu vnútorného stavu. Vstupná brána by mala chrániť vnútorný stav pred odchýlkami spôsobenými nerelevantným vstupom, na druhú stranu výstupná brána má za úlohu chrániť ostatné jednotky v sieti pred zbytočnými informáciami v pamäti.

7.6 Rekurentné konvolučné sieťe

Rekurentná konvolučná sieť (RCNN) v svojej arechitektúre kombinuje vlastnosti CNN a RNN. Cieľom je využiť silné stránky oboch typov sietí na riešenie úloh, ktoré sú vhodné pre jednu alebo druhú architektúru. Napríklad, RNN sa hodia na spracovanie časových radov a na zachytávanie dlhodobých závislostí medzi vstupmi a výstupmi, zatiaľ čo CNN sú účinné pri vyhľadávaní vzorov v obrazoch a používajú sa často na úlohy klasifikácie obrazov.

V tejto architektúre neurónových sietí sa kombinujú vlastnosti oboch, tak aby sa dosiahli lepšie výsledky. Napríklad pozitívne vlastnosti RNN sa používajú na zlepšenie CNN. Liang a Hu popisujú kombinovanú architektúru siete na detekciu objektov v jednej zo svojich prác a podobnú architektúru použili aj na označenie scény [55]. Vo svojich prácach túto kombinovanú architektúru označujú pojmom RCNN. Nasledujúci citát popisuje ich hlavnú myšlienku:

„A prominent difference is that CNN is typically a feed-forward architecture while in the visual system recurrent connections are abundant. Inspired by this fact, we propose a recurrent CNN (RCNN) for object recognition by incorporating recurrent connections into each convolutional layer“ [55]

Základnou stavebnou jednotkou RCNN je rekurentná konvolučná vrstva, ktorej skratka je RCL. Táto vrstva v sebe kombinuje prvky konvolučných a rekurentných neurónových sietí. RCL sa používajú na to, aby sa zachytili dlhodobé závislosti medzi vstupmi a výstupmi v sieti, ako sú schopné urobiť RNN. Ale tiež sú

navrhnuté tak, aby využili výkon konvolučných vrstiev na vyhľadávanie vzorov v dátach.

Obr. 7.8: Dôležitosť kontextových informácií pri rozpoznávaní nosu alebo úst [55].

Rekuretná konvolučná vrstva dokáže teda vyhľadávať vzory v dátach a zároveň zachytávať dlhodobé súvislosti v nich.

Pri detekcii objektov na vizuálnych dátach záleží na okolí, teda kontexte. Napríklad dôležitosť kontextových informácií pri úlohe rozpoznávania nosu alebo úst spočíva v tom, že tieto časti ľudského tela sa nachádzajú v konkrétnom kontexte v rámci tváre ako celku a v rámci celého obrázku. Kontextové informácie poskytujú dôležité súvislosti, vďaka ktorým môžeme lepšie pochopiť a rozpoznať jednotlivé časti tváre. Napríklad, bez znalosti kontextu, môže byť ťažké rozpoznať ústa alebo nos v izolácii od ostatných častí tváre a okolia. Ako je možné vidieť na obrázku 7.8.

Pre klasické CNN je riešenie tejto úlohy problematické. Avšak kľúčovým modulom sú opakujúce sa konvolučné vrstvy (RCL) v RCNN, ktoré do konvolučnej vrstvy zavádzajú opakujúce sa spojenia. S týmito spojeniami sa „CNN“ môže vyvíjať v čase, aj keď vstup je statický a každá jednotka je ovplyvnená svojimi susednými jednotkami.

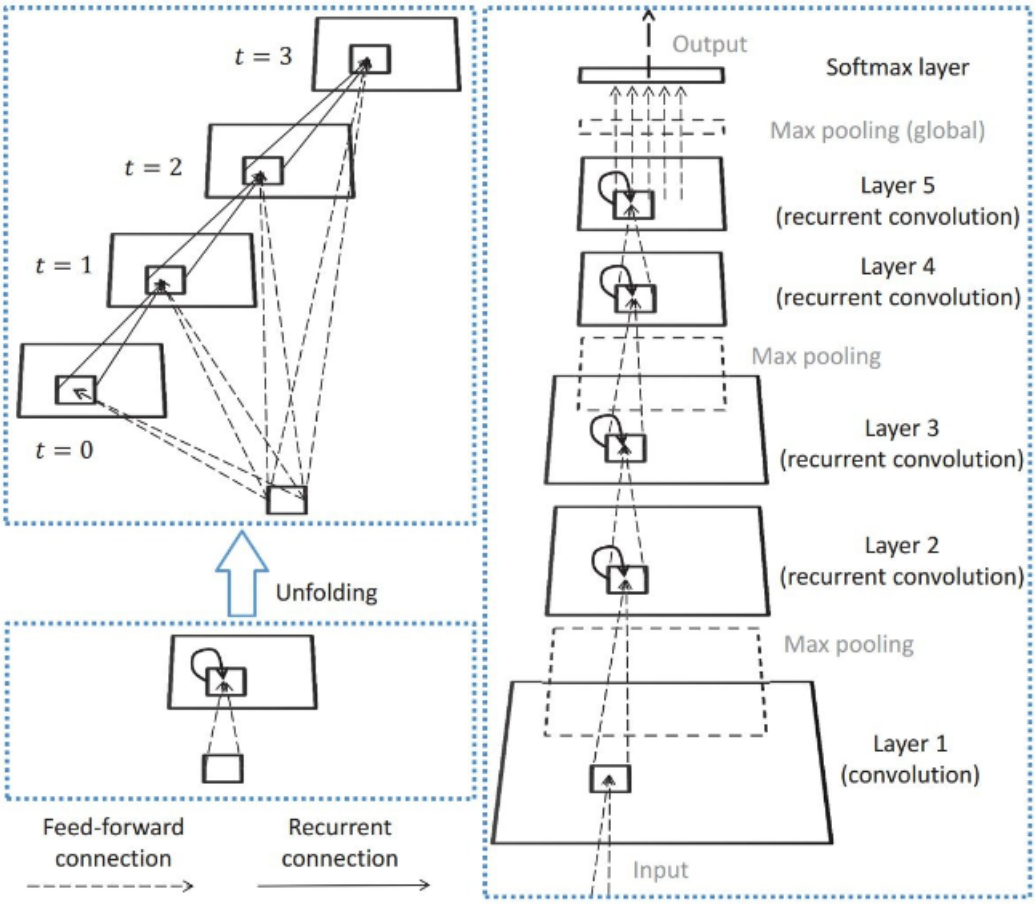
Táto vlastnosť integruje kontextové informácie o obrázku, čo je dôležité pre detekciu objektov. Dôležitosť kontextových informácií je znázornená na obrázku 7.8. Na tomto obrázku je veľmi ťažké rozpoznať ústa alebo nos bez kontextu [55].

Na obrázku 7.9 je znázornená schéma RCNN, v ľavej časti ilustrácie je znázornená rekurentná konvolučná vrstva počas troch časových krokov, to smeruje na doprednú neurónovú sieť s hĺbkou štyri.

Na pravo je zobrazené RCNN použité v práci Recurrent Convolutional Neural Network for Object Recognition s jednou konvolučnou vrstvou, štyrmi RCL, tromi max poolingovými vrstvami a jednou softmaxovou vrstvou.

Tréning týchto umelých neurónových sieti prebieha algoritmom Backpropagation through time (BPTT), ktorý sa používa pre učenie rekurentných neurónových sietí (RNN). Tento algoritmus učenia vychádza z pôvodného algoritmu backpropagation, ktorý sa používa na učenie klasických neurónových sietí, ale muselo dôjsť k modifikáciám aby bol schopný pracovať s rekurentnými vrstvami. Tento algoritmus rozdelí vstupnú časovú radu do jednotlivých krokov a pre každý krok vypočíta chybu medzi skutočným výstupom a očakávaným výstupom.

Potom sa chyba „backpropaguje“ cez sieť a používa sa na aktualizáciu váh v rekurentných vrstvách, aby sieť bola schopná lepšie predikovať výstupy aj na



Obr. 7.9: Schématická vizualizácia rekurentnej konvolučnej siete [55].

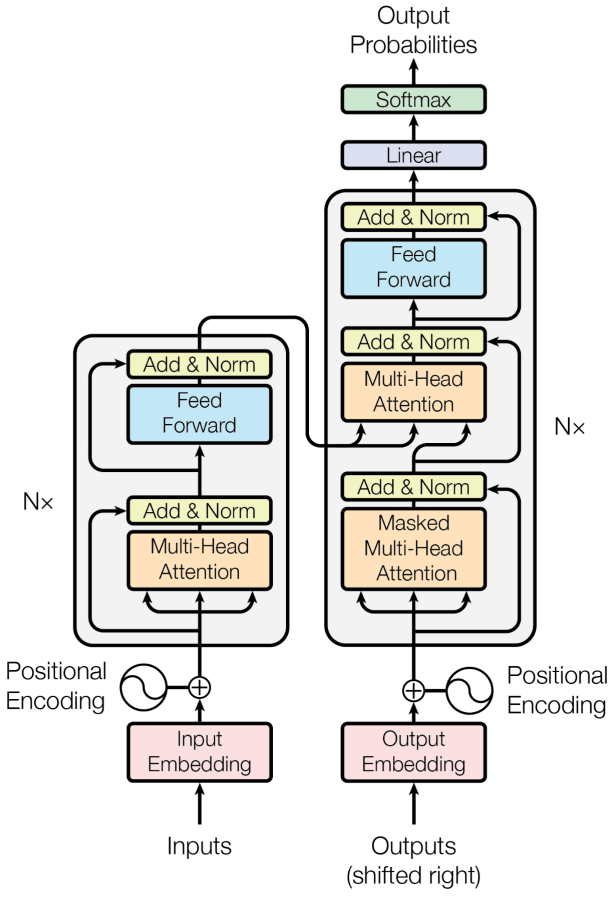
základe časovej postupnosti.

RCNN podľa výsledkov Liang a Hu, ktoré uvádzajú vo svojich prácach dosahujú lepšie výsledky ako CNN pri spracovaní vizuálnych dát, hlavne v oblasti detekcie objektov a porovnateľné alebo mierne lepšie výsledky ako RNN [55]. Ďalšou ich výhodou oproti CNN je, že sú menej náchylné na pretrénovanie. Ďalej uviedli, že tieto siete vo svojej podstate napodobňujú biologické fungovanie mozgu a preto by na nich mal byť v blízkej budúcnosti kladený väčší dôraz, Na základe týchto výsledkov sme sa rozhodli zakomponovať RCNN do tejto práce a využiť ich na detekciu falošných správ na slovenských textoch.

8. Architektúra Transformer

V roku 2017 Vaswani a jeho kolegovia predstavili model neurónovej siete založenej na architektúre Transformer, ktorý vynikal v prekladateľských úlohách z angličtiny do francúzštiny (WMT 2014 English-to-French) a do nemčiny (WMT 2014 English-to-German). Tento model bol popísaný v článku k štúdii s názvom „Attention is All You Need“, ktorý publikovali Vaswani a jeho spoluautori [56].

Transformer je tvorený dvoma primárnymi blokmi - prvým z nich je Encoder, ktorý sa tiež nazýva v slovenskom jazyku kódovač alebo kódovací blok a druhým blokom je Decoder. Tieto dva bloky sú kľúčovými časťami architektúry Transformer. Podrobná vizualizácia tejto architektúry je zobrazená na ilustrácii 8.1. Architektúra modelu Transformer, ktorá je zobrazená na obrázku 8.1, je tvorená šiestimi kódovacími a dekódovacími blokmi zoradenými za sebou.



Obr. 8.1: Architektúra modelu Transformer [56].

.

Attention

Najdôležitejšou súčasťou modelov neuronových sietí založených na architektúre Transformer je komponenta s názvom Attention. Táto komponenta nahradila do tohto momentu používané sequence-to-sequence modely [57].

Sequence-to-sequence model pracuje tak, že sa na začiatku jednotlivo spracujú pomocou Encoder bloku všetky vstupné slová. Táto úplna informácia o vstupe, ktorá má pevne daný rozmer, je následne odovzdaná Decoder bloku, ktorý postupne generuje výstupné slová. Problém však môže nastať pri dlhších vetách alebo textoch, kde nemusí byť možné zakódovať informáciu o všetkých vstupných slovách.

Bahdanau a jeho spolupracovníci [57] prišli s novou myšlienkou v oblasti strojového prekladu, ktorá umožňuje modelu si vybrať, na ktoré slová vstupného textu sa má pozerať. Teda sa na ne zamerať pomocou bloku Attention. Týmto spôsobom mohol model kódovať každé vstupné slovo do číselnej reprezentácie a potom použiť Attention blok na výber týchto reprezentácií, ktoré sú dôležité pre výpočty. Táto myšlienka bola revolučná a priniesla nový prístup k prekladu textov pomocou neurónových sietí.

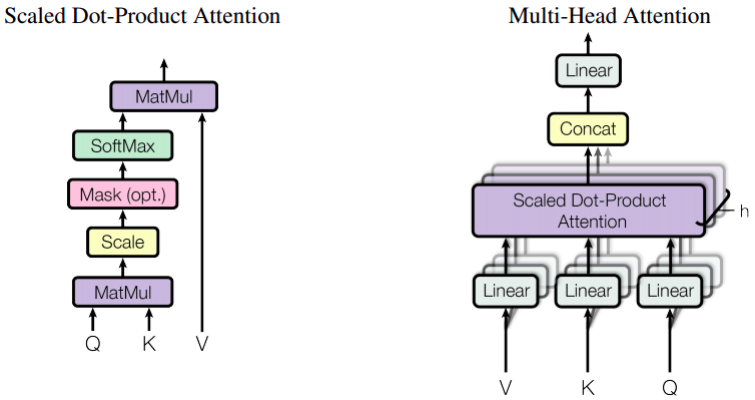
Implementácia Attention vrstvy v transformeroch

Transformery využívajú Attention bloky na to, aby mohli pri spracovávaní vstupných textov sústrediť svoju pozornosť na relevantné časti. Attention vrstva v transformeroch pracuje s tromi typmi vektorov. Ich názvy sú key (kľúč), value (hodnota) a query (požiadavka).

Výstup z bloku sa počíta ako vážený súčet values, kde váhy zodpovedajú kompatibilite odpovedajúceho key a query [56]. Pre každé vstupné slovo sa najskôr spočíta dvojica key-value. Následne počas iterácie výpočtu si model vyžiadá query ako požiadavku pre daný krok.

V ďalšom kroku je uskutočnené porovnanie query s každým key. Týmto porovnaním získame skóre pre daný key. Na výpočet skóre sa používa skalárny súčin, pretože ako key tak aj query sú číselné vektory s rovnakou dimenziou.

Posledným krokom je transformácia skóre na váhy pomocou softmax funkcie. Táto funkcia zabezpečí, že hodnoty skóre budú v intervali $\langle 0,\ 1\rangle$ a budú v súčte dávať 1, ale so zachovaním rovnakých relatívnych rozdielov medzi jednotlivými hodnotami skóre. Nové values získané pomocou softmax funkcie určujú, ako dôležitá je daná value v procese výpočtu výstupu modelu.



Obr. 8.2: Výpočet Attention v transformeri [56]. Lavá časť ilustruje, ako prebieha výpočet Attention query (Q), key (K) a value (V). Pravá časť obrázka ukazuje, ako funguje Attention s väčším počtom hláv.

V publikácii pomenovali tento princíp výpočtu Attention Scaled Dot-Product Attention, pretože využíva skalárny súčin. Proces, ktorý sme opísali vyššie, predstavuje výpočet jednej Attention hlavy v architektúre Transformer. Všimnite si, že jeden blok Attention môže obsahovať viacero takýchto hláv. Zvýšenie počtu hláv v Attention bloku pridáva len všetkým týmto výpočtom a vektorom v jednotlivých krokoch o dimenziu viac. Na obrázku 8.2 je ilustrácia výpočtu Attention v architektúre Transformer.

Self-Attention

Nápad na vytvorenie a použitie Attention vrstvy v modeloch neuronových sieti, ktoré slúžia na spracovanie textu, prišiel pri riešení prekladateľských úloh. Prvotná verzia Attention potrebovala pre svoje správne fungovanie jednak údaje zo vstupu modelu tak aj z jeho výstupu. Bolo potrebné zaistiť aby model mohol vidieť nielen vstupné slová, ale aj slová, ktoré už sám vygeneroval. Takýto mechanizmus sa odlišuje od mechanizmov Self-Attention používaných v dnešnej architektúre Transformer.

Modely neurónových sieti založené na architektúre Transformer používajú mechanizmus zvaný Self-Attention, ktorý je odlišný od pôvodného Attention. Tento mechanizmus Attention sa vzťahuje na rôzne pozície jedneho reťazca slov tak, aby vypočítal reprezentáciu tohto reťazca [56]. Self-Attention v architektúre Transformer si pre každé slovo určí pár key-value a následne v každej iterácii aj query.

Ilustrácia 8.1 znázorňuje Transformer a jeho tri rôzne Attention vrstvy. Medzi Self-Attention vrstvy patrí Attention vrstva v Encoder bloku a v Decoder bloku umiestnená dole. Posledná Attention vrstva, ktorá je znázornená na obrázku vo vnútri Decoder bloku sa líši od týchto dvoch, pretože sa zameriava na vstupné dáta pre Encoder blok. Informáciu o tom, na ktorú časť vstupu sa má zamerať, však získava zo vstupných dát pre Decoder blok. Túto vrstvu v architektúre Transformer nazývame Encoder-Decoder Attention [58].

Encoder

V neurónových sieťach založených na architektúre Transformer so vstupom do siete ako prvý pracuje Encoder blok. Tento blok sa skladá z dvoch vrstiev, prvou z nich je Self-Attention vrstva. Na ňu nadväzuje druhá časť, ktorou je klasická dopredná neurónová sieť (feed-forward).

Reziduálne spoje sú viditeľné na ilustrácii 8.1 a obklopujú obe komponenty Encoder bloku. Tieto spoje zabezpečujú, že pôvodná vstupná informácia sa propaguje do nasledujúcich blokov. Nakoniec sa na konci nachádza normalizačná funkcia s názvom Layer normalization.

Výstup získaný pomocou aktuálneho Encoder bloku je potom odoslaný na vstup do nasledujúceho Encoder bloku. Ak sa však jedná o posledný Encoder blok, potom je jeho výstup posunutý na vstup do všetkých Decoder blokov.

Decoder

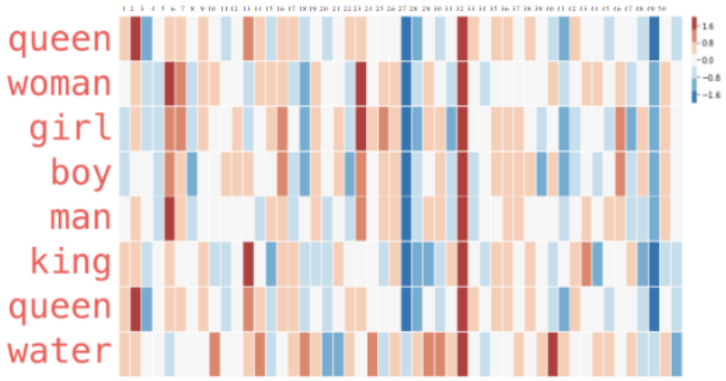
Decoder blok má tri časti, ktoré zahŕňajú Self-Attention vrstvu a doprednú neurónovú sieť, podobne ako v prípade Encoder bloku. Avšak medzi tieto dve

vrstvy je vložená Encoder-Decoder Attention vrstva, ktorej úlohou je vypočítať Attention na základe výstupu Encoder bloku. To znamená, že vstupom pre Decoder blok je doterajší výstup modelu.

Decoder blok sa odlišuje ešte od Enkoderu v jednej veci. Vstupný vektor do tejto komponenty transformeru ma vždy fixnú dĺžku. To je docielené vyplnením prázdnymi slovami sprava. Následne je nutná ešte úprava prvého Attention bloku, a to tak aby sa nepozeral na nasledujúce pozície. To je docielené tak, že pred výpočtom softmax funkcie je skóre na týchto doplnených pozíciach prepísané hodnotou $-\infty$, čo spôsobí, že softmax ich ohodnotí 0.

Vstupný embedding

Pri modeli založenom na architektúre Transformer sa na vstupe nespracovávajú priamo slová alebo vety, ale vstupný text sa najskôr rozdelí na tokeny, ktoré môžu reprezentovať celé slová, ich časti alebo aj jednotlivé znaky. Tieto tokeny sú potom zakódované pomocou kodovacieho algoritmu, ktorý ich prevedie na prirodzené čísla. Jednou z možností kódovania tokenov je ich identifikačné číslo v kódovacej tabuľke.



Obr. 8.3: Príklad GloVe vstupného embeddingu na anglických slovách [59].

Využitie vstupného embeddingu na reprezentáciu slov umožňuje, aby slová s podobným významom mali podobné vektory. Táto podobnosť je dosiahnutá tým, že každé slovo je reprezentované vektorom numerických hodnôt. Model siete sa naučí túto konkrétnu reprezentáciu slov počas trénovania.

Pri použití architektúry Transformer pre spracovanie textu je potrebné najskôr v prvom kroku ešte upraviť tento kód jednotlivých tokenov pomocou one-hot kódovania, pretože model neprijíma kód tokenu priamo, nevie s ním pracovať. One-hot kódovanie funguje tak, že pre danú veľkosť slovníka vytvorí vektor s nulami na všetkých pozíciách a na pozícii, ktorá zodpovedá kódu tokenu v slovníku je umiestnená hodnota jedna. Pre príklad, ak chceme zakódovať token s kódom 2 a veľkosť slovníka je 5, výsledný vektor bude mať tvar [0 0 1 0 0].

Vstupný embedding je vlastne tabuľka, ktorá každému tokenu priradí jeho konkrétny vektor hodnôt. Technicky, je implementovaný ako matica o dimenziách $s \times d$, kde „s“ reprezentuje veľkosť kódovacieho slovníka a „d“ je dimenzionalita modelu t.j. požadovaná veľkosť vstupu. V prípade BERT-base modelu to je matica

30522 × 768. Daný token teda bude ohodnotený 768 rôznymi zložkami, ktoré sa model naučil pre dané slovo pri tréningu.

Na ilustrácii 8.3 je ukázaný príklad vstupného embeddingu GloVe. Ak pozornejšie preskúmame tento vstupný embeding môžeme vidieť, že všetky podstatné mená sa zhodujú v 31.zložke, ktorá je reprezentovaná bordovou zložkou. Táto značka teda nesie informáciu o slovnom druhe vstupného slova. Za povšimnutie stojí aj značka číslo 26, všetky životné podstatné mená majú v tejto zložke tmavomodrú hodnotu, iba water (voda) sa od týchto slov odlišuje. Z toho vyplýva, že táto časť nesie informáciu o životnosti objektu, ktore slovo popisuje. Ďalšie informácie, ktoré môžeme vyčítať s vizualizácie sú slová rovnakého pohlavia ako sú napríklad man (muž) a boy (chlapec) majú podobné niektoré úseky, na druhú stranu slová ako king (kráľ) a queen (kráľovná) sú podobné v iných úsekoch.

Pozičné kódovanie

Ďalšou dôležitou súčasťou architektúry Transformer je pozičné kódovanie, ktoré umožňuje modelu brať do úvahy aj poradie jednotlivých vstupných tokenov. To je veľmi dôležité, pretože v modeloch neurónových sieti založených na architektúre Transformer sa nepoužívajú konvolučné ani rekurentné vrstvy, ktoré by zabezpečovali poradie tokenov. Pozičné kódovanie sa pripočítava k vstupnému embeddingu a zabezpečuje, aby pozícia každého tokenu bola zakódovaná do modelu. Bez tohto kódovania by model nedokázal rozlišovať poradie slov vo vete. Táto myšlienka bola rovnako prvýkrát predstavená v článku Vaswani a kolektív v roku 2017 [56].

Každá pozícia vo vstupnom texte má svoj jedinečný vektor, ktorý sa vytvára pomocou pozičného kódovania. Pri výpočte tohto vektora sa využívajú goniometrické funkcie sínus a kosínus. Pozičné kódovanie je kľúčové pre zachytenie poradia slov vo vete, ktoré by inak nebolo zohľadnené v architektúre Transformer.

Pre párné pozície sa používa funkcia sínus a pre nepárné kosínus. Vzorec na výpočet i-tej zložky slova na pozícii *pos* je definovaný pomocou tejto funkcie:

$$PE_{pos,2_i} = \sin\left(\frac{pos}{10000^{\frac{2_i}{d_{model}}}}\right), \tag{8.1}$$

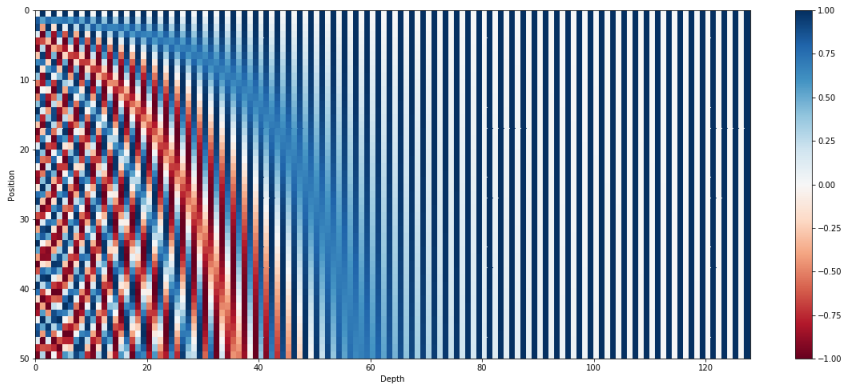
$$PE_{pos,2_{i+1}} = \cos\left(\frac{pos}{10000^{\frac{2_i}{d_{model}}}}\right), \tag{8.2}$$

kde parameter d_{model} vyjadruje dimenzionalitu modelu.

Výstup

Architektúra Transformer modelu zahŕňa výstupnú vrstvu, ktorá sa nachádza na konci Decoder bloku a pozostáva z plne prepojenej vrstvy neurónov. Táto vrstva produkuje skóre pre každý token. Na výstup Decoder bloku sa pripája softmax vrstva, ktorá prevedie skóre na pravdepodobnosti pre každý token. Tento výstup je finálnym výstupom jednej iterácie architektúry Transformer. Potom sa tento výstup použije ako vstup Decoder bloku na generovanie ďalšieho tokenu v ďalšom kroku.

Na ilustrácii 8.4 je použité pozičné kódovanie s dimenziou $d_{model} = 128$ pre 50 začiatočných pozícií vstupu. Táto funkcia je špeciálne navrhnutá tak, aby sa



Obr. 8.4: Pozičné kódovanie pre prvých 50 pozícií, $d_{model} = 128$ [60].

model mohol naučiť pracovať s relatívnou pozíciou vstupných tokenov. Pretože každý pevný posun o k v pozícii sa dá vyjadriť ako lineárna kombinácia PE_{pos+k} a PE_{pos}, kde PE_{pos} je pozičné kódovanie pre pozíciu pos.

8.1 BERT

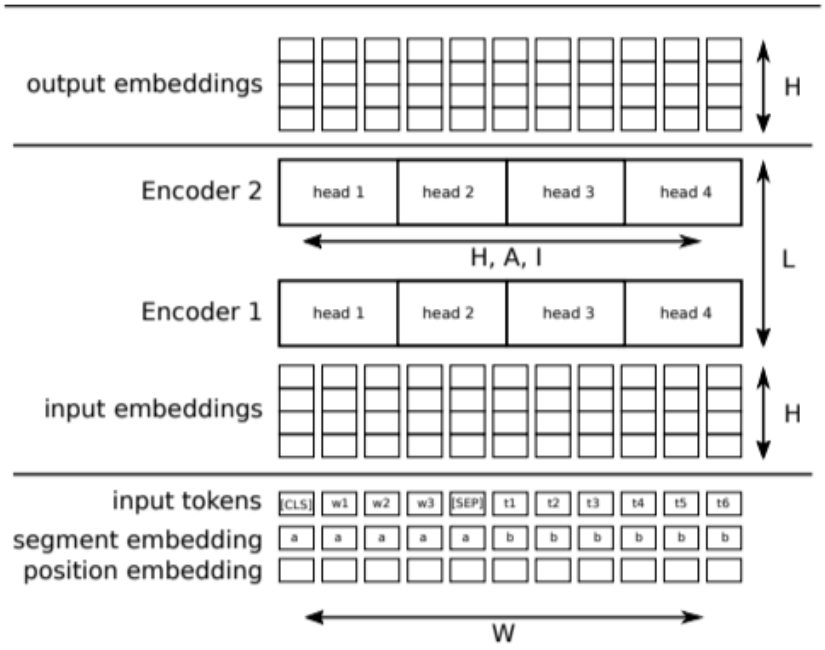
V tejto časti si bližšie predstavíme BERT, hlavný model, ktorý budeme používať v tejto práci. BERT je skratka pre Bidirectional Encoder Representations from Transformers, je to jazykový model, ktorý využíva aplikáciu obojsmerne učeného transformeru (mechanizmus pozornosti). Tento model je založený na architektúre Transformer a bol navrhnutý Devlin a kol. v roku 2018 [61].

BERT je založený na myšlienke transferového učenia. Najprv sa tento model musí natrénovať na veľkej množine počiatočných dát, ale potom sa dá relatívne rýchlo doladiť a prispôsobiť na riešenie širokej škály úloh. Aj keď je proces predtrénovania modelu náročný, je nutné ho spraviť len raz a predtrénovaný model je verejne dostupný.

8.1.1 Architektúra modelu

Architektúra modelu BERT je založená na Encoder bloku z Transformeru. V nasledujúcej časti zadefinujeme jednotlivé parametre BERT modelu, ktoré korešpondujú s veľkosťami BERT-base modelu.

- V – veľkosť slovnej zásoby (30 000)
- W – maximálna dĺžka vstupnej sekvencie (512)
- L – počet vrstiev Encoderu (12)
- H – hidden size, rozmer key, value, query pre Attention v Encoderi (768)
- I – intermediate size, rozmer doprednej neuronovej siete Encodera (3072)
- A – počet Attention hláv (12)



Obr. 8.5: Schéma architektúry modelu BERT vrátane parametrov [61].

8.1.2 Tokenizátor

BERT používa tokenizér WordPiece [62]. WordPiece je jednoduchý tokenizér, ktorý používa greedy algoritmus na rozdelenie každého slova na tokeny. Jeho slovná zásoba má dve časti: začiatočné tokeny (celé slová alebo predpony) a nasledovné tokeny (prípony a infixy). Napríklad slovo *flying* je rozdelené na tokeny *fly* a *ing*.

Vzhľadom k tomu, že WordPiece je interný nástroj spoločnosti Google, vydali iba proces tokenizácie a nie proces tvorby slovnej zásoby. Z tohto dôvodu boli vývojari nútení pre nové modely vytvoriť svoje vlastné súbory so slovnou zásobou.

8.1.3 Vstup

Vstupné tokeny sú najprv zakódované pomocou one-hot kódovnia a potom premietnuté do „a“ vektorov dimenzie H. Tento vstupný embedding je totožný ako v architektúre Transfomer, inicializované sú náhodne a naučené počas predtrénovania.

Prvý vstupný token je vždy [CLS]. Kvôli kontextualite modelu môže jeho embedding obsahovať informácie o celej vstupnej sekvencii. Embedding [CLS] sa zvyčajne používa v klasifikačných úlohách.

Často je dôležité vedieť, v ktorej časti sekvencie sa konkrétny token nachádza. Na tento účel používa BERT pozičné kodovanie ako paralelný vstup do tokenov. Rovnako ako transfomer, viac v kapitole 8.

Niektoré úlohy NLP, ako napríklad odpovedanie na otázky, vyžadujú, aby bol model schopný akceptovať viacero sekvencií ako iba vstup štýlu (<„Otázka“, „Odpoveď“>). Architektúra BERT to rieši dvoma spôsobmi. Prvým je použitie špeciálneho tokenu [SEP] na oddelenie sekvencií. Druhým je embedding segmentov. Embedding segmentu je paralelný vstup k tokenom a ukazuje, do ktorej

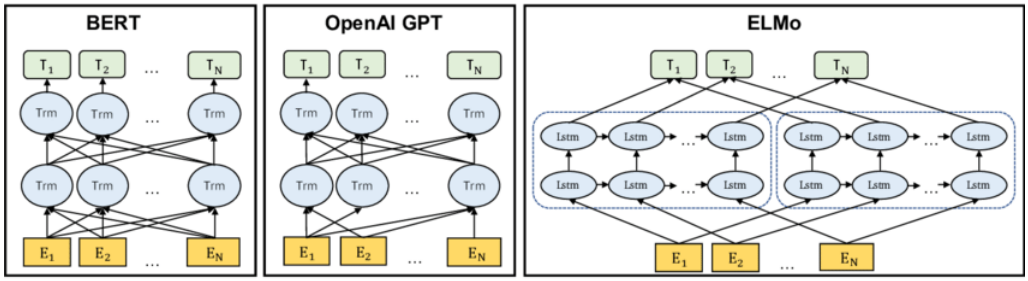
sekvencie token patrí. Toto embedding sa učí počas trénovania modelu.

Výsledny embedding je súčtom všetkých troch vyššie uvedených embeddings (vstupný, pozičný, segmentový). Ten sa následne posunie na vstup do Encoder bloku. Každý výsledny vektor je stale iba výsledkom pre jeden vstupný token. Rozmer tenzora embeddingu vstupu je dimenzie (W, H).

8.1.4 Encoder vrstvy

Každá z vrstiev Encodera vykonáva multi-head attention na každom tokene rovnakým spôsobom ako v Transformere. Tento proces vytvára kontextový embedding. Existuje L vrstiev Encodera, každá z nich má A hlav, ktoré zdieľajú spolu H neurónov. Každá hlava má H/A neurónov (dimenzie query, key a value) v prípade BERT-base je to 64. Dopredná neurónova sieť, ktorá je v Encoder bloku má veľkosť každej Attention vrstvy rovnú hodnote I.

Rovnako ako pri Transformeri sa na všetky vstupné embeddings aplikujú rovnaké váhy. Rozmer tensora, ktorý vystupuje z Encoder bloku je dimenzie (W, H).



Obr. 8.6: Porovnanie architektúr BERT, GPT A ELMo [61].

Na rozdiel od ELMo, ktorý má zreteľné prechody zľava doprava a sprava doľava, alebo GPT [63], ktorý má iba ľavý kontext, má BERT realný obojsmerný kontext.

8.1.5 Predtrénovanie

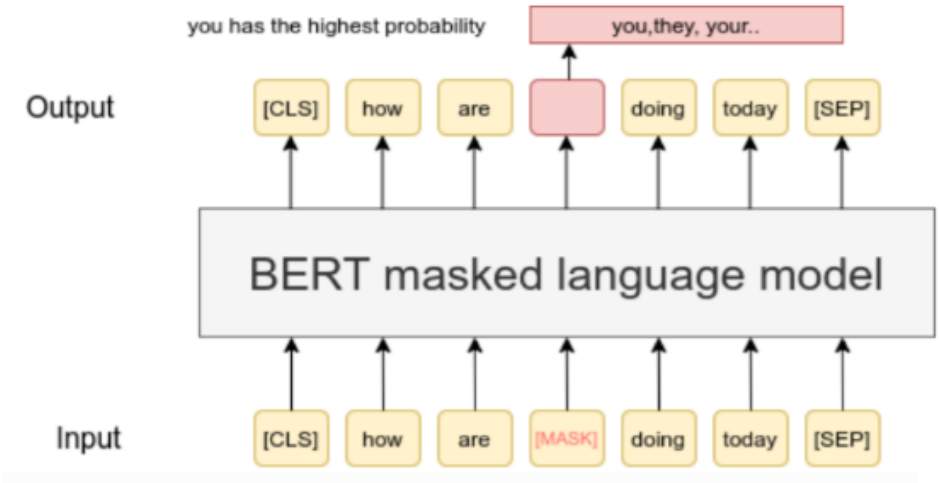
Autori navrhli dve samostatné úlohy na predtrénovanie, ktoré sa dajú automaticky vygenerovať z korpusu. Týmito predtrénovanými úlohami sú Masked Language Modelling a Next Sentence Prediction. Následné prispôsobenie modelu pre iné úlohy sa vykonáva pripojením ďalších vrstiev na základnú časť modelu.

Masked Language Modelling

BERT sa líši od tradičných jazykových modelov tým, že sa pri trénovaní snaží predpovedať maskované tokeny vstupnej sekvencie. Pre tento účel využíva techniku masked language model (MLM), pri ktorej sa 15% tokenov zo vstupu náhodne zamaskuje. Model sa následne snaží klasifikovať zamaskovaný token len na základe kontextu. Tokeny sú nahradené ich vektorovou reprezentáciou pomocou techniky WordPiece.

Sekvencia tokenov musí začínať špeciálnym tokenom [CLS] a končiť špeciálnym tokenom [SEP]. Maskovaný token sa zvyčajne nahradí tokenom [MASK].

V prípade, že sa token maskuje, slovo sa nahradí iným v 10% a v 10% zostáva nezmenené. Cieľom modelu je správne predpovedať pôvodný token iba na základe obojsmerného kontextu. V diagrame na obrázku 8.7 je tento postup zobrazený graficky.



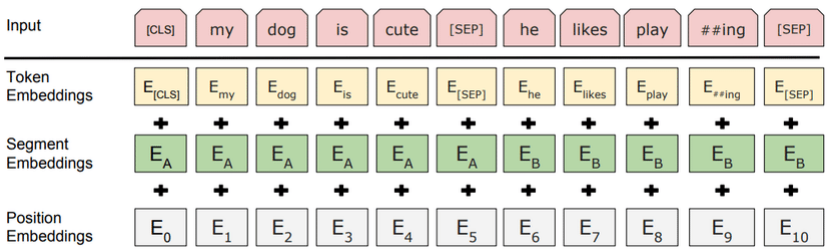
Obr. 8.7: Masked Language Modelling [64].

Next Sentence Prediction

Druhou hlavnou úlohou, ktorú BERT rieši je úloha NSP. Toto je obzvlášť užitočné pre všetky iné úlohy založené na porozumení vzťahu medzi dvoma vetami (napr. odpovede na otázky alebo odvodenie z prirodzeného jazyka).

Túto úlohu môžeme binarizovať tak, že vzhľadom na dvojicu viet A a B trénujeme model na pochopenie toho, či ide o po sebe idúce vety alebo nie. Pri danej dvojici viet je cieľom tejto úlohy rozhodnúť, či druhá veta bola v pôvodnom texte hneď po prvej. Na zodpovedanie tejto úlohy model používa kontextové vloženie tokenu [CLS] (ktorý je vždy prvým vstupným tokenom) v binárnom klasifikátore.

Napríklad v ideálnom prípade model neklasifikuje vety „The child plays in the park..“ a „The catcher in the rye is a great book.“ ako po sebe idúce vety.

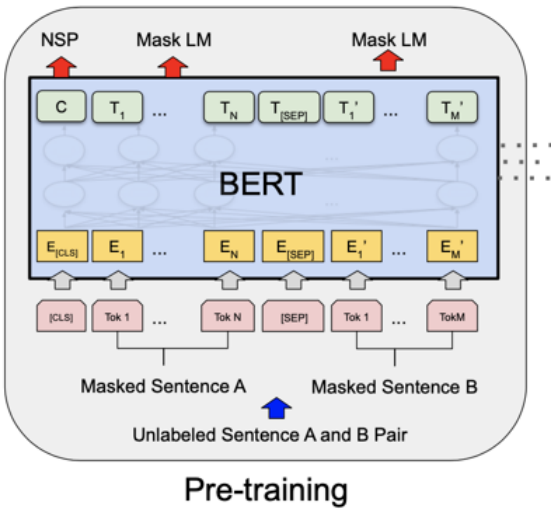


Obr. 8.8: Reprezentácia vstupného embeddingu pre model BERT. Vyjadrená ako súčet tokena, segmentu a pozičného kódovania [61].

Pre obe úlohy používa BERT separátnu stratovú funkciu v podobe cross-entropy. V predtréningu je výsledná stratová funckia (loss) vyjadrená jednodu-

cho pomocou lineárnej kombinácie týchto dvoch. Aby sme však tieto úlohy splnili, musíme mierne upraviť vstupné embeddings generované napríklad pomocou vopred trénovaného WordPiece tokenizéra. Robíme to pridaním embeddingu pozície (z dôvodu vysvetleného v predchádzajúcich častiach) a embeddingu segmentu do každého embeddingu tokenu. Toto sa robí s cieľom poskytnúť BERT informáciu, či token patrí do prvej alebo druhej vety pre úlohu NSP.

Obrázok 8.8 zobrazuje vstupnú reprezentáciu, ktorá vstupuje do BERT modelu. Okrem toho dodávame našim vstupom dva špeciálne tokeny: token [CLS] označujúci začiatok sekvencie a token [SEP], ktorý označuje koniec vety, t. j. umiestnený medzi párom viet a na konci celej sekvencie. Finálny embedding tokenu [CLS], skratka pre klasifikačný token, ako už naznačuje jeho názov, sa používa na konečnú klasifikáciu v rámci klasifikačnej úlohy. V skutočnosti, ako experimenty Clarka a kol. (2019) ukazujú [65], že väčšina informácií, t. j. entropia, obsiahnutá vo výstupnej sekvencii, je zahrnutá v tomto embeddingu tokenu, ktorý preto možno považovať za určitý druh embeddingu viet. Takže združovacia vrstva na obrázku 8.8 nemá byť zamýšľaná ako zvyčajná združovacia vrstva (ako je maximálne alebo priemerné združovanie), ale skôr ako vrstva, ktorá združuje, t. j. extrahuje, iba finálny embedding [CLS]. To sa potom odovzdá plne pripojenej vrstve, ktorá po aktivácii klasifikuje výstup pomocou sigmoidnej funkcie v našom kontexte viacerých značiek.



Obr. 8.9: Reprezentácia predtréningovej metódy pre model BERT s celou jeho štruktúrou, ako [CLS] a [SEP] token, vrátane MLM a NSP [61].

Obrázok 8.9 sumarizuje predtréningový postup so všetkými jeho hlavnými charakteristikami. Vo svojej pôvodnej verzii je BERT trénovaný na BooksCorpus (asi 800 miliónov slov) a anglickej Wikipédii (asi 2500 miliónov slov). Trénovaci proces zabral približne 40 epoch.

V čase, keď bol tento model predstavený, BERT prekonal všetky už existujúce LM v rôznych úlohách a stal sa de facto najmodernejším modelom NLP. Viac podrobností o BERT modele možno nájsť v Devlin et al. (2018) [61].

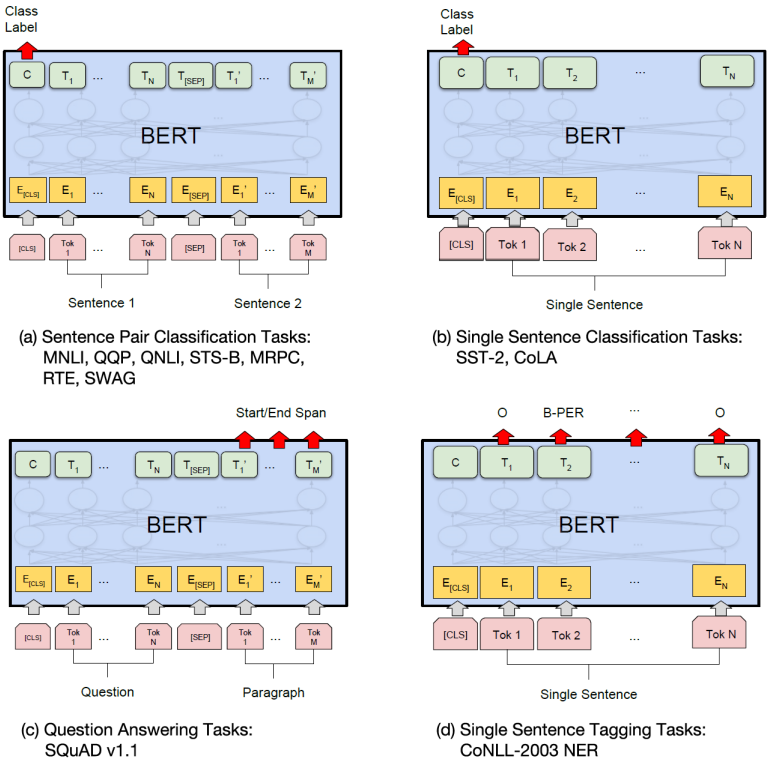
8.1.6 Fine-tuning

Fine-tuning je proces prispôsobenia predtrénovaného modelu BERT na inú úlohu, ako na ktorú bol predtrénovaný.

Prvým krokom je úprava vstupu úlohy tak, aby vyhovovala vstupom modelu BERT a trochu pripomínala predtréningové úlohy. To je možné uskutočniť pri mnohých úlohách, pretože BERT podporuje viacero vstupných sekvencií. Avšak kvôli svojej obojsmernej povahe a pevnej vstupnej dĺžke sekvencii nie je BERT vhodný na úlohy generovania textu (zatiaľ čo GPT je [63]).

Druhým krokom je pridanie ďalšej vrstvy (vrstiev) na koniec posledného Enconder bloku. Tieto novo pridané vrstvy by mali byť schopné odpovedať na novo zadanú úlohu. To zvyčajne znamená pridanie klasifikačnej vrstvy na výstupný embedding prvého tokenu ([CLS]), ak chceme klasifikovať celý vstup (návrh, analýza sentimentu), alebo pomocou všetkých embeddings výstupných tokenov inými spôsobmi špecifickými pre úlohu na úrovni tokenov (označovanie sekvencií, zodpovedanie otázok).

Posledným krokom je trénovať takto upravený model BERT na nových dátach, ktoré sú špecifické pre novo zadanú úlohu. Počas tohto procesu sa trénujú všetky váhy modelu. Príklady možných úloh, ktoré sa takýmto spôsobom dajú riešiť zahŕňajú všetky úlohy GLUE [66], klasifikáciu textu, sekvenčné označovanie, parafrázovanie, zodpovedanie otázok a mnohé ďalšie.



Obr. 8.10: Fine-tuning modelu BERT na oboch úlohách. [61]

Proces fine-tuningu je oveľa rýchlejší ako tréning, Devlin a kol. uvádzajú až 300x zrýchlenie. Kde svoj experiment porovnávali 16 TPU počas 4 dni versus 1 TPU po dobu 1 hodiny [61].

8.2 Modifikácie modelu BERT

BERT má veľa rôznych variantov a v tejto časti spomenieme tie najpodstatnejšie, ktoré v tejto práci budú implementované a porovnané v rade experimentov, vzhľadom na ich úspešnosť pri detekcií falošných správ na slovenskom datasete.

8.2.1 mBERT

Viacjazyčný BERT je rozšírením pôvodného modelu BERT. Tento model má zhodnú architektúru s BERT model popísanú v sekcii 8.1, ale je vopred natrénovaný na 102-104 najväčších Wikipédiách vrátane tej slovenskej. V súčasnosti sú dostupné dve verzie, prvá je cased, ktorá má WordPieces vrátane veľkých a malých písmen. Druhá verzia modelu je viacjazyčná uncased, ktorá má všetky podslová písané malými písmenami a bez diakritiky. Prekvapivé je, že aj pre jazyky, ktoré majú len malé percento trénovacích príkladov, to v skutočnosti funguje veľmi dobre. Pre objasnenie, len jedno percento tréningových príkladov je vo viacjazyčnom BERT pre slovenský jazyk.

Výhodou je, že bez akéhokoľvek explicitného dohľadu je mBERT schopný reprezentovať viac ako sto vstupných jazykov v zdieľanom priestore. Vo vete mBERT chápe, že slovo dog a slovo pes majú rovnaký význam bez toho, aby v príkladoch v tréningu videl konkrétny slovník alebo konkrétne informácie. Táto reprezentácia viacjazyčného jazyka v rovnakom zdieľanom priestore nám umožňuje vykonávať takzvaný medzijazyčný prenos.

Výsledky dátového súboru Multilingual Question Answering (MLQA) pre úlohu čítania s porozumením sú publikované v článku Lewis a kol. (2019) [67]. V tejto úlohe je potrebné v odseku umiestniť otázku a odpoveď. Údaje sú dostupné v 7 rôznych jazykoch. Trénovanie modelu v angličtine a následné spustenie v inom jazyku funguje porovnateľne s prekladom údajov do angličtiny a potom späť.

8.2.2 RoBERTa

Tento model neurónovej siete založený na architektúre Transformer, optimalizuje a zrobustňuje pôvodný model BERT. Prvýkrát bol predstavený Liu a kol. v roku 2019, ako modifikácia architektúry BERT [68].

Next sentence prediction, ktorú sme popisali v sekcii 8.1. V pôvodnej verzii modelu BERT bola zamýšľaná ako veľmi dôležitá časť tréningu modelu BERT. Tento fakt naznačovali experimenty, ktoré vykonal Devlina a kol. pri pôvodnom návrhu architektúry [61].

Neskoršie experimenty však ukázali, že odstránenie tejto úlohy z trénovania modelu môže viesť k zlepšeniu výsledkov. Preto autori modelu RoBERTa vykonali nasledujúce experimenty:

- Segment-pair: pár segmentov s celkovým maximálnym počtom tokenov 512
- Sentence-pair: pár viet prirodzeného jazyka, ktoré sú zvyčajne kratšie ako 512 tokenov.
- Full-sentences: Iba jeden segment ako vstup s 512 tokenmi, môže prekročiť hranicu dokumentu.

- Doc-sentences: Iba jeden segment ako vstup s 512 tokenmi, nemôže prekročiť hranicu dokumentu.

Výsledky jednotlivých experimentov sú zhrnuté v článku „Roberta: A robustly optimized bert pretraining approach“, ktorý publikovali Liu a kol. [68]. Výsledky sú v tomto článku prehľadne uvedené v tabuľkách. Z výsledkov, ktoré boli publikované v tomto článku vyplynulo, že full-sentences dosahuje pomerne dobrých výsledkov v porovnaní s ostatnými prístupmi.

Z tohto dôvodu je RoBERTa trénovaná s dynamickým maskovaním celých viet bez predikcie ďalšej vety s 8 tisíc minibatches a BPE na úrovni bajtov (Byte-Pair Encoding) s 50 tisícmi proslov. Pri zachovaní pôvodnej architektúry BERT modelu, ktorá je znázornená v sekcii 8.1 na obrázku 8.5.

8.2.3 SlovakBERT

V tejto časti predstavíme model strojového učenia nazvaný SlovakBert. Tento model je založený na architektúre Transformer, konkrétne na type RoBeRTa a bol trénovaný na veľkom množstve dát v slovenskom jazyku. Tento model vytvorili KInIT v spolupráci s Gerulata Technologies s cieľom zlepšiť automatické spracovanie slovenských textov [69].

Za posledné roky neurónové jazykové modely zaznamenali najvýznamnejší pokrok v oblasti spracovania prirodzeneho jazyka (NLP). Pomocou týchto jazykových modelov dokázali výskumníci zlepšiť výsledky v mnohých úlohách spracovania textu a slúžia ako technologický základ aj v takých aplikáciách ako Google Search alebo Google Translate, ktoré denne používajú miliardy ľudí.

Neurónové jazykovedné modely spočiatku vznikali najmä pre angličtinu a následne pre veľké jazyky, ako čínština, francúzština, s odstupom času pribudla napr. aj čeština a poľština. Dostupné sú dokonca aj multilingválne modely, ktoré sme spomenuli v predošlých sekciách.

SlovakBert je prvým takýmto modelom, ktorý je založený na architektúre Transformer, určeným pre slovenský jazyk. Tento model bol vytvorený koncom roka 2021, a publikovaný v článku „SlovakBERT: Slovak Masked Language Model“ [69]. SlovakBert si rozoberiem podrobnejšie vzhľadom k jeho výhodam, ktoré vyplývajú z jeho príslušnosti k slovenskému jazyku.

Architektúra modelu

Architektúra je založená na RobERTa [68], ktorá je už v základe robustnejšia ako u BERT-a. V tabuľke 8.1 je uvedený detailný popis tejto architektúry. Ako tokenizér bol použitý BPE [70] s veľkosťou abecedy 50264.

Tréning modelu

Učenie modelu prebiehalo na hardvéri tvorenom 4 NVIDIA A100 GPUs, a zabralo aproximovane 248 hodín. Model bol trénovaný približne 300k training steps s veľkosťou batch size 512, čo je približne 70 epoch. Každá epocha pozostáva z 4277 trénovacích krokov. Ako optimalizér bol použitý optimalizačný algoritmus Adam s hodnotou učiaceho koeficientu with 5×10^{-4}.

SlovakBERT	
Architecture	RoBERTa
Num. layers	12
Num. attention head	12
Hidden size	768
Num. parameters	125M
Languages	1
Training dataset size (tokens)	4.6B
Slovak dataset size (tokens)	4.6B
Vocabulary size	50K
Universal Dependencies train set tokenization	
Average token length (chars)	3.23
Average word length (tokens)	1.43
Effective vocabulary	16.6K
Effective vocabulary (%)	33.05

Tabuľka 8.1: Architektúra modelu SlovakBERT [69].

Regularizácia učenia bola docielená Dropout = 0.1 a weight decay (λ = 0.01).Dĺžka vstupných sekvencii bola limitovaná na 512 tokenov so snahou zachovať, čo najviac sekvencií v plnom formáte.

Trénovanie modelu SlovakBERT nebolo jednoduchou úlohou. Samotný tréning vyžadoval takmer dva týždne výpočtov na serveri s výkonným hardvérom. Ak by sme to porovnali s bežným počítačom s grafickou kartou strednej triedy, trénovanie by trvalo roky, a s bežným pracovným notebookom dokonca aj desaťročia.

Trénovacie dáta

Ako trénovací súbor dát bola použitá kombinácia dostupných dátových súborov v slovenskom jazyku a dataset získaný Web-crawling slovenských webových stránok. Dostupné súbory dát v slovenskom jazyku, ktoré boli použité pre tréning sú:

- Wikipedia (326 MB textu)
- Open Subtitles (415 MB)
- OSCAR 2019 dataset (4,6 GB)

Tieto datasety boli rozšírené o dáta, ktoré boli získané prehľadaním webových stránok s topdoménou *.sk*. Na jednotlivé webové stránky bola aplikovaná detekcia jazyka a boli z nej extrahované iba názov a čistý textový obsah bez HTML značiek. Veľkosť takto získaných dát bola 17,4 GB. Text bol potom spracovaný nasledujúcimi krokmi:

- Adresy URL a e-mailové adresy boli nahradené špeciálnymi symbolmi.

- Opakujúce sa interpunkčné znamienka boli redukované, t.j. ak existovali sekvencie tvorené rovnakým interpunkčným znamienkom, potom boli zredukované iba na jedno znamienko (napr. !!!! na !).
- Markdown syntax bola odstránená.
- Všetok textový obsah v zátvorkách typu .... bol odstránený, aby sa eliminovalo množstvo značiek a programovacieho jazyka.

Výsledný dátový súbor bol rozdelený na vety a boli z neho odstránené duplicity. Finálny dataset tvorí 181,6 milióna unikátnych viet. Tieto dáta tvoria obraz toho, čo model považuje za slovenčinu. Celkovo má konečný dátový súbor veľkosť 19,35 GB.

Evaluácia modelu

Výsledný model bol otestovaný výskumníkmi z KInIT na rozličných úlohách z oblasti spracovania prirodzeného jazyka. SlovakBEERT dosahuje výborné výsledky pri gramatickej analýze, semantickej analýze, rozpoznávaní sentimentov či klasifikácii dokumentov.

Výsledky jednotlivých experimentov sú dostupné vo verejne dostupnom článku „SlovakBERT: Slovak Masked Language Model“ [69]. V súčasnosti je SlovakBert jedným z najlepších modelov pre spracovanie slovenského jazyka. Natrénovaný model je verejne dostupný, môže ho teda NLP komunita využiť na riešenie rôznych problémov spracovania prirodzeného jazyka v slovenčine.

9. Fake-news datasety

Získanie dostatočne veľkého dátoveho súboru pre klasifikáciu falošných správ a dezinformácií je náramne náročný proces. Týmto náročnym krokom získania dátoveho súboru to však iba začína, jeho následné spracovanie a klasifikovanie je ešte komplikovanejšie.

V dnešnej dobe, existuje iba obmedzené množstvo dátových súborov, ktoré sú k dispozícii pre tento fenomén v porovnaní s ostatnými klasifikačnými problémami. To je spôsobené náročnosťou tvorby datasetov pokrývajúcich problém fake-news. Problém pri tvorbe týchto datasetov spočíva v tom, že neexistuje dostatočne kvalitný automatický systém na anotáciu. Z tohto dôvodu je nutné všetky datasety manuálne anotovať. Každý jeden text anotuje viac anotátorov, aby sa zabránilo chybovosti a zaujatosti jednotlivca. Tento postup, ktorým sa vytvárajú datasety zaoberajúce sa témou fake-news, je náročný na čas aj ľudské zdroje (anotátorov). S tým je spojený ďalší problém pre kvalitnú anotáciu je potrebné väčšie množstvo anotátor to môže byť problém pre menšie jazyky, kde týchto anotátorov môže byť nedostatok. Príkladom je aj slovenský jazyk, ktorý má jedného oficiálneho anotátora pre Facebook kontent.

Z týchto dôvodov sa väčšina prác zaoberajúcich fake-news venovala jazykom s väčším jazykovým korpusom, teda s dostatkom dát, ako sú angličtina, nemčina a francúzština. Pre slovenský jazyk sa nám podarilo získať momentálne jednu z prvých neverejných dátových sád, na ktorej experimentovali Sarnovský s tímom [2]. Vzhľadom k nedostatku kvalitných dát v slovenskom jazyku sa pokúsime v experimentálnej časti tejto práce využiť aj cudzojazyčné datasety pokrývajúce tému fake-news. S cieľom transformovať metódy úspešné na jazykoch s dostatkom dát na slovenčinu.

V nasledujúcich podkapitolách predstavíme dátové sady, ktoré boli použité v tejto práci. Zameriavame sa na ich pôvod, jazyk, základné štatistiky a vlastnosti jednotlivých textov. Datasety sú rozdelené do dvoch skupín anglickej a slovenskej podľa jazyka, v ktorom sú texty v datových sadách.

9.1 Anglické dátové sady

Pre anglický jazyk existuje niekoľko dátových sád, pre naše účely sme si vybrali dve z nich. Prvým je dataset s názvom LIAR, ktorý je známy v NLP komunite. Ako druhý sme zvoli COVID19 FN, dôvodom voľby tohto datasetu je charakter slovenskej dátovej sady, ktorú sa nám podarilo získať. Slovenská dátová sada je špecificky zameraná na covid-19 fake-news články. Chceme teda porovnať aj vplyv témy článkov a úspešnosť transferu medzi jazykmi. Z tohto dôvodu boli zvolené práve tieto dva datasety.

9.1.1 LIAR

LIAR je datová sada, pozostávajúca z výrokov a novinových článkov, ktoré sú anotované. Dátová sada bola vytvorená výskumníkmi Univerzity v Kalifornii v Berkeley a má slúžiť na tréning a hodnotenie modelov, ktoré dokážu rozpoznať falošné správy [6].

Dátová sada obsahuje celkovo 12.8k manuálne anotovaných novinových článkov z rôznych zdrojov, vrátane politickych blogov, stránok overujúcich fakty a hlavných zdrojov správ. Podrobná správa a analýza jednotlivých článkov bola vykonaná pomocou *politifact.com*, kde každý článok analyzovali a overili jeho zdroj.

V dobe vytvorenia bol najväčším datasetom svojho druhu, kedy obsahoval 12.8k vyhlásení, ktoré boli manuálne označené človekom pomocou API na stránke *politifact.com*, následne bol ešte každý článok vyhodnotený editorom z pravdivostného hľadiska.

LIAR dataset je užitočným zdrojom dát pre výskumníkov a odborníkov, ktorí pracujú na vývoji techník na detekciu falošných správ. Môže byť použitý na tréning a hodnotenie algoritmov strojového učenia a tiež na získanie poznatkov o charakteristikách falošných správ a o spôsoboch ich šírenia.

Každý článok je označený jednou zo šiestich kategórií: $pants-fire$, $false$, $barely-true$, $half-true$, $mostly-true$ a $true$. Dátová sada obsahuje tiež ďalšie metadáta pre každý článok, ako je zdroj a dátum publikovania.

Dataset vo formáte ako bol zverejnený je rozdelený do troch disjunktných množín, trénovacej, validačnej a testovacej množiny. V tabuľkách 9.1 a 9.2 je možne vidieť distribúciu príkladov do jednotlivých množín.

Dataset Statistics	
Train set size	10 269
validation set size	1284
Test set size	1283
Avg. statement length (tokens)	17.9

Tabuľka 9.1: LIAR štatistiky dátovej sady [6].

Autori článkov	
Democrats	4150
Republicans	5687
None(FB posts)	2185

Tabuľka 9.2: LIAR štatistiky autorov článkov [6].

Túto datovu sadu sme si zvolili pre jej značnú popularitu v zahraničnej komunite NLP pri detekcii fake-news. Hlavný dôvodom však je téma, ktorou sa zaoberajú jednotlivé texty. LIAR pokrýva najmä politické témy, ktoré súviseli s voľbami v USA v roku 2016. Články v datasete LIAR majú diametrálne odlišnú tému ako tie obsiahnuté v slovenskom datasete (covid-19). Tento fakt nám dáva priestor na overenie domienok, že fake-news majú podobný formát naprieč rôznymi témami. Prípadne túto domienku pre slovenský jazyk vyvrátiť.

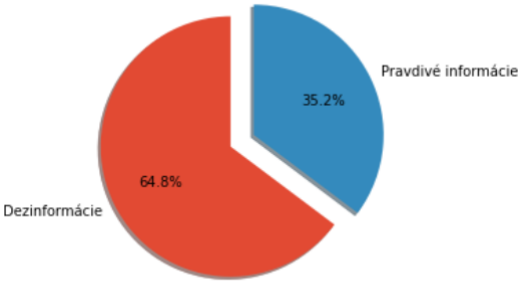
Predspracovanie dátovej sady

Pre účely tejto práce je nutné upraviť pôvodnú dátovú sadu LIAR. Z charakteru klasifikačných tried v cieľovom slovenskom datasete je nevyhnutné upraviť

klasifikačné triedy aj v datasete LIAR. Z pôvodných 6 tried dôjde k redukcii na binárny priznak, kde 1 značí falošnú správu a 0 pravdivú. Pôvodné príznaky tried sme namapovali na toto nové rozloženie nasledovne. Klasifikačné triedy „mostly-true“ a „true“ sme spojili do novej triedy, ktorá charakterizuje pravdivú množinu s príznakom 0. Ostatné klasifikačné triedy z pôvodného datasetu sú v novej dátovej sade chápané ako nepravdivé t.j. 1.

V dátovej sade došlo ešte k jednej zmene, pôvodné rozloženie článkov do troch množín train,valid a test sme zmenili. V novej dátovej sade train a valid tvoria jednu trénovaciu množinu a pôvodná test-množina sa používa na validáciu. Tento krok sme vykonali z dôvodu zväčšenia trénovacej množiny, vzhľadom k tomu že evaluácia sa bude diať na slovenskom datasete.

Po tejto úprave je rozdelenie do tried 64,8% nepravdivých spáv a 35,2% pravdivých ako je možné vidieť na obrazku 9.1.



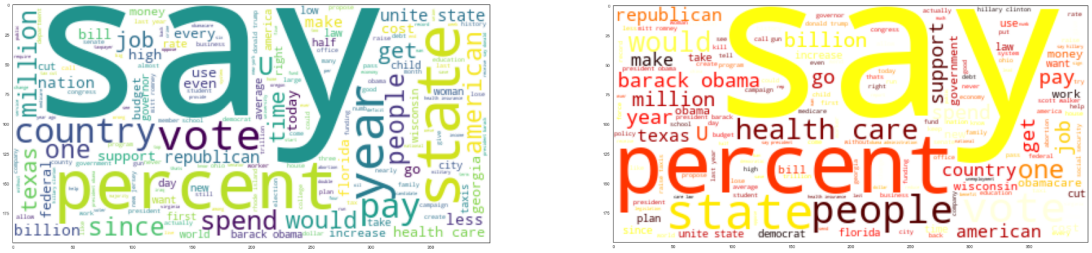
Obr. 9.1: Distribúcia do tried upraveného LIAR.

Vizualizácia dátovej sady

V tejto časti sú vizualizácie LIAR datasetu pre oboznámenie sa s charakterom dát, ktoré ho tvoria.

Na nasledujúcich vizualizáciach sú zobrazené World clouds pre jednotlivé cieľové množiny. Pri tomto zobrazení veľkosť slova odzrkadľuje jeho význam v jednotlivých textoch t.j. veľkosť tokenov odzrkadľuje ich frekvenciu v dátach.

Obr. 9.2: LIAR: WordCloud pre všetky články.



(a) Pravdivé články (b) Falošné články

Obr. 9.3: LIAR: WordCloud pre pravdivé a falošné články.

Na vizualizáciach dominujú slovné spojenia z oblasti volieb a s tým spojených politikov a problémov. Tento výstup je očakávaný, keďže LIAR dataset pokrýva práve túto oblasť fake-news,

9.1.2 COVID19 FN

Covid19 FN je ďalším datasetom ktorým sa budeme zaoberať v tejto práci [3]. Tento dataset bol zvolený z dôvodu rovnakej témy článkov ako ma slovenský dataset t.j. články spojené s pandémiou covid-19. V experimentálnej časti tejto práce porovnávame výsledky s modelmi natrénovanými na datasete LIAR 9.1.1. Toto porovnanie nám potvrdí alebo vyvráti dôležitosť témy článkov pri detekcii fake-news.

Covid19 FN je dataset, ktorý pôvodne vytvorili Parth a kol. [3]. Tento dataset obsahuje správy týkajúce sa ochorenia covid-19 zozbierané z rôznych webov na kontrolu faktov ako sú *Politifact.com*, *NewsChecker*, *Boomlive* a z nástrojov ako Google fact-check-explorer a IFCN chatbot.

Táto dátová sada pozostáva z tweetov, článkov a príspevkov na socialnych sieťach. Pri tvorbe datasetu najskôr identifikovali správy súvisiace s pandémiou covid-19. Následne manuálne detekovali jazyk správ, do datasetu boli zahrnuté iba správy, ktoré boli písané v anglickom jazyku. Následne rozdelili manuálne správy na pravdivé a falošné.

Pravdivé správy boli zozbierané z overených zdrojov, ktoré pojednávajú o ochorení Covid-19. Ako sú stránky a účty vládnych organizácii, zdravotnícke organizácie a noviny. Konkrétne boli zozbierané zo 14 rôznych zdrojov ako sú World Health Organization (WHO), Centers for Disease Control and Prevention (CDC), Covid India Seva, IndianCouncil of Medical Research (ICMR), a ďalšie. Každá správa bola manuálne anotovaná človekom, ktorý určil jej pravdivosť a relevantnosť k téme covid-19.

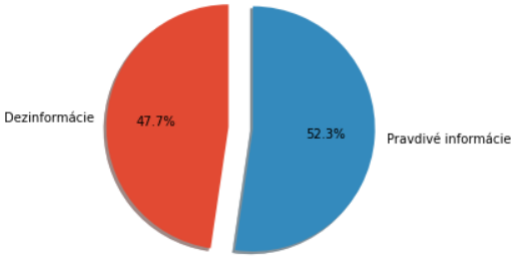
Falošné správy sú zozbierané zo špekulatívnych zdrojov, ktoré boli verifikované ako fake-news. Rovnako ako pravdivé správy bolo manuálne overené, či daná správa súvisí s covid-19. Anotácia pravdivostnej hodnoty prebehla manuálnym sposobom t.j. hodnotiteľom bol človek.

Dátová sada vo forme ako bola publikovaná je rozdelená do troch disjunktných množín a to trénovacej, validačnej a testovacej množiny. Distribúcia správ do jednotlivých množín je zobrazená v tabuľke 9.3.

Dataset Statistics	
Train set size	6420
validation set size	2140
Test set size	2140
Avg. statement length (tokens)	27.05

Tabuľka 9.3: COVID19 FN: štatistiky dátovej sady [3].

V tejto datovej sade nie je potrebná úprava klasifikačných tried, pretože sa zhodujú s triedami v slovenskom datasete. Dataset je rozdelený do 3 podmnožín, ktoré sú vybalansované rovnako ako celý dataset. Tento fakt veľmi napomáha učeniu modelov a nie je nutné daný dataset augumentovať.



Obr. 9.4: Distribúcia do tried COVID19 FN.

Predspracovanie dátovej sady

V dátovej sade sme rovnako ako pri LIAR 9.1.1 datasete zmenili pôvodné rozloženie článkov y troch množín na dve. V novej dátovej sade train a valid tvoria jednu trénovaciu množinu a pôvodná test množina sa používa na validáciu. Tento krok sme vykonali z rovnakých dôvodov ako v datovej sade LIAR.

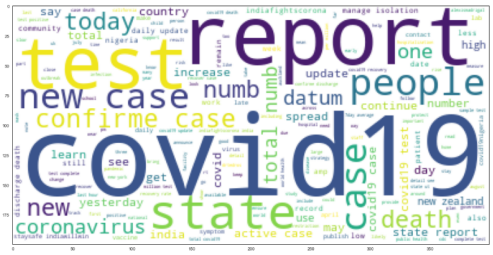
Na druhú stranu, dataset pozostáva primárne z príspevkov na socialnych sietiach, obsahuje aj nealfanumerické znaky ako sú napríklad rôzne emoji a sprievodné znaky. Tieto znaky bolo nutné zo správ vymazať z dôvodu, že veľké lingistické modely s týmito symbolmi nevedia pracovať a iba by to prinieslo šum do dát. Tieto symboly boli z datasetu odstránené.

Vizualizácia dátovej sady

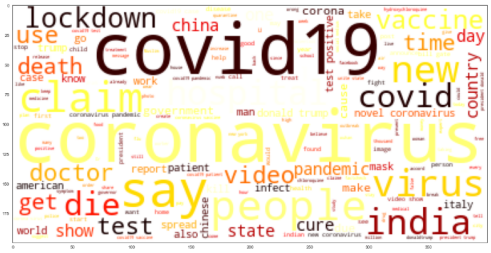
V tejto časti sú vizualizácie datovej sady COVID19 FN pre oboznámenie sa s charakterom dát, ktoré ho tvoria. Analyzovali sme tiež dataset na úrovni tokenov. 5 najčastejších tokenov po odstránení stopových slov:

- Falošné: coronavirus, covid19, people, report, new
- Skutočné: covid19, case, news, tests, report
- Kombinované: covid19, case, coronavirus, news, people

Obr. 9.5: COVID19 FN: WordCloud pre všetky články.



(a) Pravdivé články



(b) Falošné články

Obr. 9.6: COVID19 FN: WordCloud pre pravdivé a falošné články.

Ilustrácie 9.5 a 9.6 vizualizujú WordClouds frekvenciu slov pre jednotlivé klasifikačné triedy. Z vizualizácií pomocou word clouds a manuálnej analýzy vyplynulo, že medzi falošnými a skutočnými správami dochádza k výraznému prekrývaniu dôležitých slov. Tento jav je spôsobený hlavne tým, že obe spadajú pod jednu tému pandémia covid-19.

9.2 Slovenské dátové sady

Verejné dátové sady určené pre úlohu klasifikácie fake-news v slovenskom jazyku momentálne nie sú dostupné. Existuje ich pár ale zatiaľ žiaden repozitár nie je verejne dostupný. Z tohto dôvodu sa tejto problematike venuje minimum vedcov z oblasti NLP na Slovensku.

Tento stav nie je spôsobený absenciou fake-news na Slovensku, ale problémom s anotáciou textov potrebných pre vytvorenie takéhoto datasetu.

Nám sa v tejto práci podarilo získať dataset vytvorený Sarnovským a kol. [2], ktorý nie je aktuálne verejne dostupný. Za poskytnutie tohto datasetu sme mu veľmi vďačný, položil tým základný kameň celej tejto práce. Na základe dát z tohto datasetu sme vytvorili v tejto práci nový menší dataset, ktorý je popísaný v sekcii 9.2.2.

Tvorcovia majú v pláne v budúcnosti publikovať celý dataset, čo by mohlo rozbehnúť automatickú klasifikáciu fake-news v slovenskom jazyku.

9.2.1 SlovakCovid19 FN

Táto dátová sada je prvým slovenským datasetom, ktorý v tejto práci predstavíme. SlovakCovid19 FN vytvoril Sarnovský a kol. na Technickej univerzite v Košiciach [2]. Táto dátová sada nie je verejne dostupná a je nutné si o ňu požiadať.

SlovakCovid19 FN je dataset, ktorý je jedným z prvých slovenských datasetov pre úlohu klasifikácie fake-news. Táto dátová sada obsahuje správy týkajúce sa ochorenia covid-19 zozbierané z rôznych slovenských webov. Overenie pravdivostnej hodnoty článkov najskôr prebehlo automaticky kedy boli jednotlivé zdrojové weby ohodnotené podľa relevantnosti na *Konspiratori.sk*. Na základe tohto ohodnotenia boli články z jednotlivých webov rozdelené do skupín na pravdivé a nepravdivé. Táto pôvodná myšlienka sa ale ukázala ako nesprávna. Pretože aj stránky, ktoré šíria dezinformácie občas zverejňujú pravdivú správu. Táto skutočnosť priniesla do dát množstvo nesprávne ohodnotených článkov.

Z tohto dôvodu bol zavedený druhý level automatického overenia pravdivostnej hodnoty článkov, ktorý sa zameriaval na určenie pôvodného zdroja správy. Týmito dvoma krokmi boli dáta ohodnotené automaticky.

Paralelne s týmto postupom sa jednotlivé články hodnotili aj manuálne za pomoci 5 anotátorov. Týmito anotátormi boli študenti na univerzite. Výsledná anotácia datasetu bola vytvorená spojením manuálnej a automatickej anotácie.

Dataset je tvorený pravdivostnou hodnotou a obsahom článku. Kontent jednotlivých článkov v dátovej sade tak ako nám bola poskytnutá netvoria čisté textové dáta, ale podľa zdroja aj rôzny šum. Medzi tento šum môžeme zaradiť HTML tagy, url odkazy na multimediálny obsah, hlavičky stránok, metadáta webu atď. Tento šum však nie je pre všetky články v dátovej sade rovnaký. Medzi jednotlivými článkami sa odlišuje podľa zdrojoveho webu. Preto nie sú tieto dáta vhodné pre použitie v klasifikácii pre všetky články.

Aby sme mohli tento dataset použiť pre klasifikáciu za pomoci NLP bolo nutné ho od týchto dodatočných informácii očistiť.

Predspracovanie dátovej sady

V prvom kroku bol tento dataset očistený od netextových dát, ktoré by prekážali pri klasifikácii pomocou textových dát, z datasetu teda boli odstránené:

- Adresy URL a e-mailové adresy boli odstranené.
- Opakujúce sa interpunkčné znamienka boli redukované, t.j. ak existovali sekvencie tvorené rovnakým interpunkčným znamienkom, potom boli zredukované iba na jedno znamienko (napr. !!!! na !).
- Markdown syntax bola odstránená.
- Všetok textový obsah v zátvorkách typu .... bol odstránený, aby sa eliminovalo množstvo značiek a programovacieho jazyka.
- Hlavičky článkov.
- Všetok nealfanumerický balast.

Počas predspracovania textových dát sme narazili aj na zaujímavé vlastnosti niektorých zdrojov, ktoré z týchto dát boli čitateľné.

Prvou z takýchto vlastnosti je zdroj: Hlavné Noviny, väčšina článkov z toho zdroja v sebe obsahovala text na konci, ktorým vyzývali na finančnú podporu tohto „nezávislého“ media v boji proti vláde a úradom. Tento text dopĺňa informáciu o obsahu, ktorý dané medium šíri, preto sme ho ponechali v datasete. Pre Hlavné Noviny, to nekončí iba pri tejto jednej zaujímavosti. Ďalšou bola informácia v pätičke webovej stránky, ktorá bola pri web-crawlingu stiahnutá z ich webu. V tejto časti webovej stránky sa zriekli všetkých následkov v prípade šírenia nimi publikovaných článkov. Plus informovali ako ich vláda a iné nezávislé médiá na Slovensku chcú „umlčať“.

Poslednou zaujímavou vlastnosťou, ktorá sa ukázala v dátach bola informácia o autoroch niektorých konšpiračných teórii. Týchto článkov bolo v jednotkách, ale zaujímavé bolo, že všetky tieto články podpísané týmto autorom boli fake-news.

Infomácie o autorovi boli vo väčšine prípadov vymazené manuálne z textových dát aby sa výsledný NLP klasifikator nezameriaval iba na tieto konkrétne časti textov, ale aby sa zameral na kontext daného článku.

Na druhú stranu, pomocou týchto zistení by sa dali skonštruovať ďalšie umelé metadáta k textovým a postupne vybudovať zložitejší dataset, tento dataset by obsahoval ako textové dáta, tak aj kontext k nim v podobe metadát. Následne by sa tento dataset dal použiť na natrénovanie Ensemble systémov alebo iných kombinovaných klasifikátorov. Ale vzhľadom k malému množstvu dát s týmito informáciami to pre tento dataset nebolo použiteľné.

Výsledkom tohto predspracovania bol slovenský dataset obsahujúci iba relevantné textové dáta.

Štatistiky	
Falošné články	851
Pravdivé články	12885

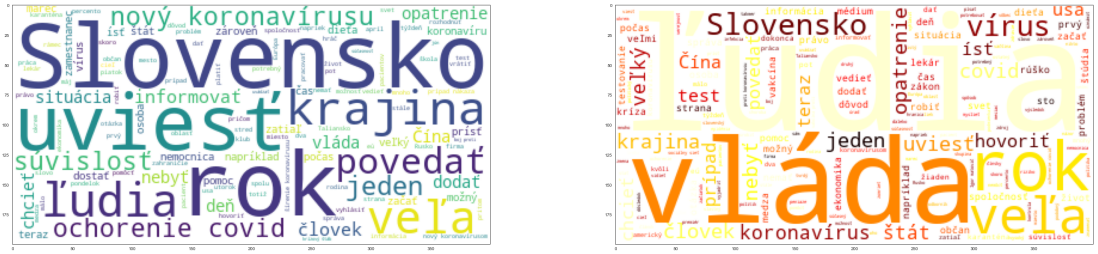
Tabuľka 9.4: SlovakCovid19 FN : štatistiky datovej sady.

Vizualizácia dátovej sady

V tejto časti sú vizualizácie datovej sady SlovakCovid19 FN pre oboznámenie sa s charakterom dát, ktoré ho tvoria.

Obr. 9.7: SlovakCovid19 FN: WordCloud pre všetky články.

Z ilustrácii 9.7 a 9.8, ktoré zobrazujú Word cloud pre slovenský dataset. Je možné vyčítať informácie o prebiehajúcej pandémii, pre ktorú rezonujú výrazy v každom z obrázkov. Podobnosť celkového k pravdivým článkom je spôsobená prevahou týchto dát v datasete. V týchto článkoch je prevaha informácií o pandémii covid-19. Na druhú stranu, vo Word cloud pre fake-news si môžeme všimnúť mierne odlišnosti oproti predošlým vizualizáciam. V týchto článkoch viac rezonujú pojmy spojené s vládou, organizáciami a opatrenia. Akosi sa strácajú informácie o prebiehajúcej pandémií a viac sa články sústredia na štátne orgány a politiku.



(a) Pravdivé články (b) Falošné články

Obr. 9.8: SlovakCovid19 FN FN: WordCloud pre pravdivé a falošné články.

9.2.2 DownSampled SlovakCovid19 FN

K vytvoreniu tohto datasetu sme pristúpili, potom ako sme sa oboznámili s pôvodným datasetom SlovakCovid19 FN, ktorý nám poskytli Sarnovsky a kol. [2]. Nový dataset, je vytvorený ako základná slovenská dátová sada pre klasifikáciu fake-news v téme padémie covid-19.

Tento dataset má slúžiť ako odrazový mostík pre NLP modely detekujúce fake-news na Slovensku. Týmto datasetom nemáme za cieľ nahradiť pôvodný dataset, práve naopak má ho dopĺňať. Práca s týmto datasetom je jednoduchšia, keďže je už očistený od šumu (HTML značky, hlavičky stránok, url, odkazy na multimedialne dáta atď.). Táto korektúra zmenšuje časovú náročnosť predspracovania textu pri jeho budúcich využitiach.

Cieľom tohto datasetu je oboznámenie s problematikou fake-news na Slovensku, a reprezentuje úvodný krok pre prácu s pôvodným datasetom SlovakCovid19 FN, s ktorým je práca ťažšia ale po získaní skúsenosti na nami vytvorenom datasete. Je možné informácie získané z tohto datasetu uplatniť aj na riešenie problému v pôvodnej verzii datasetu.

Poslednou dôležitou informáciou, ktorú musíme spomenúť. Dataset sme vytvorili vybalansovaný a to z dôvodu lepšieho porovnávania s vybalansovanými cudzojazyčnými datasetmi, ktoré sú verejne dostupné. Na tomto nami vytvorenom datasete sme vykonali radu testov, ktoré sú uvedené v kapitolách o experimentoch.

Tvorba datového súboru

Tento dataset je vytvorený z dátovej sady poskytnutej Sarnovským a kol. [2], ktorí vytvorili pôvodnú verziu datasetu pre klasifikáciu fake-news na slovenských

textoch. V prvom kroku bol tento dataset očistený od netextových dát, rovnako ako pri predspracovaní textových dát pre dataset SlovakCovid19 FN. Z datasetu teda boli odstránené:

- Adresy URL a e-mailové adresy.
- Opakujúce sa interpunkčné znamienka boli redukované, t.j. ak existovali sekvencie tvorené rovnakým interpunkčným znamienkom, potom boli zredukované iba na jedno znamienko (napr. !!!! na !).
- Markdown syntax bola odstránená.
- Všetok textový obsah v zátvorkách typu .... bol odstránený, aby sa eliminovalo množstvo značiek a programovacieho jazyka.
- Všetok nealfanumerický balast.
- Texty kratšie ako 100 znakov

Po tomto kroku, kedy boli vstupné dáta očistené od všetkého šumu. Sme získali čisté textové dáta, tvorené článkami z pôvodného datasetu.

Po tomto očistení nasledovala samotná tvorba datasetu. Prvým krokom tvorby samotného datasetu bola extrakcia článkov, ktoré sú nepravdivé. Extrahovali sme všetky fake-news články, ktoré ostali po očistení z pôvodného datasetu. Žiaden článok sme nevynechali z dôvodu, že v pôvodnom datasete tvorili približne iba 6% t.j. asi 800 článkov. V novom datasete sme chceli zachovať, čo najväčší počet článkov, z tohto dôvodu sme ponechali všetky fake-news články z pôvodného datasetu, ktoré ostali po očistení od šumu (počet klesol zhruba o 100).

Druhým krokom tvorby datasetu bolo vybratie rovnako veľkej množiny pravdivých článkov z pôvodného datasetu, To sme urobili nasledovným spôsobom, vybrali sme prvú náhodnú množinu pravdivých článkov. Túto množinu sme celú manuálne prešli a overili jej pravdivostnú hodnotu. V tomto kroku boli niektoré nevhodné články odstránené. Takže ich bolo nutné nahradiť novou náhodnou množinou z dát, ktoré ostali ešte v datasete. Tento krok sme opakovali 3-krát. Overenie pravdivostnej hodnoty pôvodných článkov robili traja ľudia. Táto fáza zabrala približne 1 hodinu denne po dobu 3 týždňov.

Po dokončení tohto kroku sme vytvorili nový DownSampled SlovakCovid19 FN dataset, ktorý je tvorený iba textovými dátami a prešiel dodatočnou manuálnou anotáciou. DownSampled SlovakCovid19 FN dataset je tvorený 1466 článkami s tematikou covid-19, ktorých distribúciu do tried môžeme vidieť v tabuľke 9.5.

Štatistiky	
Falošné články	733
Pravdiné články	733

Tabuľka 9.5: DownSampled SlovakCovid19 FN : štatistiky datovej sady.

10. Non-NLP detekcia fake-news

V tejto práci sa primárne zaoberáme detekciou falošných správ na základe ich textového obsahu. Tento problém chápeme ako úlohu textovej klasifikácie a väčšina experimentov, ktoré sme vykonali a budú popísané v nasledujúcich kapitolách, sa zaoberá práve touto problematikou. Tento princíp detekcie falošných správ vyplynul z dostupných dát pre slovenský jazyk, ktoré sa nám podarilo získať. Základná sada experimentov sa predsa len zameriava na nelingvistické metódy detekcie fake-news v slovenskom jazyku, ktoré sme popísali v kapitole 3. Dataset, ktorý sme získali pre slovenský jazyk nie je primárne koncipovaný pre nelingvistické metódy detekcie falošných správ. Preto neobsahuje štruktúrované informácie o zdrojových stránkach, autorovi a priložených prílohách (obrázky, odkazy, zdroje atď.).

Vzhľadom k tomu, že sme ho získali v jeho zdrojovej a neočistenej forme, sme zdrojový súbor dát museli predspracovať a očistiť od niektorých netextových dát ako sú HTML tagy alebo nealfanumerické znaky typu smajlíky a mnohé iné. Rovnako sme odstraňovali metadáta, ktoré boli uložené spolu s textovým obsahom webovej stránky. To sme museli uskutočniť z dôvodu, že tieto dáta boli iba v podmnožine textových súborov a mohli by skresliť výsledky lingvistických metód strojového učenia.

Počas čistenia dát a preprocessingu sme pre podmnožinu textov extrahovali informácie, ktoré sú v zaujímavom vzťahu k niektorým stránkam, autorom alebo špecifickým webové odkazom, ktoré indikujú vyšší výskyt fake-news. Potom sme na základe nich potvrdili niektoré hypotézy o neligvistických metódach detekcie falošných správ aj na podmnožine datasetu pre slovenský jazyk. Výsledky týchto zistení uvedieme v tejto kapitole.

10.1 Autor textu

V rade experimentov rôznych zahraničných štúdii bola popísaná súvislosť medzi falošnými správami a ich autorom, pretože autor je typicky zodpovedný za tvorbu a šírenie nepravdivých alebo zavádzajúcich správ. Autori, ktorí majú zlý úmysel, môžu zámerne vytvárať fake-news s cieľom zmanipulovať verejnú mienku, ovplyvniť voľby alebo podnietiť konflikty. Takíto autori sa môžu snažiť vyvolávať emócie, ako sú strach, hnev alebo podozrenie, aby dosiahli svoje ciele.

Okrem toho sa však môže stať, že aj dobre zameraní autori môžu neúmyselne šíriť fake-news. To môže nastať, ak autori neoverujú informácie ktoré zdieľajú, alebo ak dôverujú zdrojom, ktoré sú nespoľahlivé alebo neoverené.

Týchto autorov spája jedno, že v ich tvorbe sa vyskytuje väčšie percento falošných správ. Tento fakt, že autor ktorý publikoval fake-news má vyššiu tendenciu publikovať tento typ správ aj v budúcnosti potvrdila rada zahraničných štúdií. Potvrdenie tejto hypotézy sme očakávali aj pre slovenský jazyk ale výsledky nás mierne prekvapili. Aj keď pri spätnom vyhodnotení a zamyslení sa nad nimi a nad veľkosťou slovenského novinového priestoru je pravdepodobné, že za väčšinu fake-news na Slovensku môže hŕstka novinárov, ktorí ich šíria.

Po očistení textových dát od HTML tagov a metadát sme uskutočnili radu vizualizačných experimentov nad týmito dátami, aby sme lepšie pochopili slovnú

skladbu týchto textov. Pri zobrazení najfrekventovanejších slov pomocou Word Cloud a následnej kalkulácii počtu týchto slov nám v podmnožine datasetu, ktorú tvoria falošné správy, vyskočilo meno *Jaroslav Zajac*. Preto sme túto skutočnosť bližšie preskúmali. Po bližšom preskúmaní sa ukázalo, že *Jaroslav Zajac* je autorom 129 článkov, z toho 124 je klasifikovaných ako fake-news. A to z celkového počtu 851 falošných správ, ktoré tvoria dataset. To znamená že každá ôsma správa ktorá je klasifikovaná ako falošná pochádza od tohto autora.

Ďalšími autormi, ktorých by sme mohli zaradiť do tejto skupiny „šíriteľov" falošných správ sú *Jana Tutková*, *Patrik Sloboda* a napríklad aj *Ján Lakota*. Napríklad *J. Tutková* má publikované 3 články, *P. Sloboda* ich má 8 a posledným autorom ktorého sme spomenuli je *J. Lakota* s 5 článkami. Títo autori majú iba jednotky článkov v našom datasete, ale všetky ich články sú klasifikované ako fake-news. Týmto ich nechceme oficiálne označiť za šíriteľov fake-news, ale v našej dátovej sade majú vyššie percento fake-news a v nej by takto označení boli. Oficiálne označenie je ale možné až na základe väčšieho množstvá dát,no ani vtedy sa to nedá tvrdiť so 100% istotou.

Na druhej strane stoja autori *Ivan Lehotský* a *Tatiana Stará*, u ktorých nemáme zaznamenaný v našom datasete žiaden falošný článok. Od prvého menovaného autora máme v datasete 52 článkov a u druhej autorky dokonca úctyhodných 142 pravdivých článkom.

Výsledkom týchto zistení pre náš dataset je potvrdenie hypotézy, že niektorí autori majú vyššiu tendenciu publikovania falošných správ, či už úmyselne alebo nie.

10.2 Zdroj textu

Ako sme uviedli v kapitole 3, automatická kontrola zdrojov by mohla byť efektívnou možnosťou detekcie falošných správ. Tento spôsob by odhalil pôvod informácie, teda či pochádza z hodnoverného zdroja. Ak by sa v reťazci zdrojov zacyklila, označila by danú informáciu za nedôveryhodnú.

Podobne ako v predošlom prípade detekcie falošných správ na základe autora, ani v tomto prípade nemáme v datasete uvedené zdrojové údaje pri všetkých článkoch ani samostatne, ani v rámci ich textovej formy. Pri tomto prípade je ťažké určiť čo je zdroj pre daný text, pretože ak nie je uvedený v hlavičke alebo niekde v určitej časti textu, ale iba ako *URL*, je to automaticky ťažšie detekovateľné. Avšak pri vizualizácii datasetov sa nám podarilo taktiež odhaliť menšiu množinu zdrojov, ktoré sú zodpovedné za šírenie falošných správ ako *NaturalNews.com*, *Badatel.net*, *rt.com*. Z týchto zdrojov čerpajú články, ktoré sú vo väčšine prípadov klasifikované ako fake-news.

Zdroj	Pravdivé články	Fake-nwes
NaturalNews.com	0	5
Badatel.net	1	29
rt.com	6	31

Tabuľka 10.1: Zdroje fake-nwes.

Ako je možné vidieť v tabuľke 10.1, falošné správy sa často opierajú o weby, ktoré majú za cieľ navodiť dojem pravdivej správy. Častokrát sú tieto stránky koncipované štýlom, že sú nezávislé média alebo sú v nevôli politických entít a orgánov, aby navodili dojem, že sú tým správnym zdrojom informácii. Majú tiež tendenciu neúmerne propagovať dodatky ako *nezávislý*, *neskorumpovaný*, *pravda* a iné slovné spojenia, ktoré majú u čitateľa podvedome navodiť pocit dôvery v toto médium, aby v ďalšom kroku túto navodenú dôveru využili na šírenie falošných správ.

Preto je pri detekcii falošných správ rovnako dôležité overiť relevantnosť autora a aj zdroje, o ktoré sa napísaný článok opiera.

10.3 Špecifický textový obsah v metadátach

Pri skúmaní textov v dátovej sade sme zistili ešte jednu zaujímavú vlastnosť. Približne polovica falošných správ v datasete obsahovala v appendixe minimálne jeden z týchto reťazcov: „Ak vás článok“, „Vyhlásenie: Názory autora sa nemusia zhodovať“ alebo „Sme pod neustálym tlakomMnohým kruhom nevyhovuje existencia Hlavných správPodporte jediné väčšie nezávislé médium na Slovensku“.

Bližším skúmaním týchto textov sme prišli k zisteniu, že všetky pochádzajú z jedného zdroja a tým boli Hlavné správy. Aj keď v sebe článok priamo neodkazoval na tento web, pri tvorbe datasetu sa spolu s hlavným telom článku stiahli aj tieto dodatočné informácie z pätičky webu. Následne pri spracovávaní sme našli 100% zhodu týchto reťazcov s webom Hlavné správy, rovnako ako ich umiestnenie v rámci článkov, ktoré na ňom publikujú.

Články, ktoré obsahovali aspoň jeden z týchto reťazcov, boli všetky klasifikované ako fake-news. Touto zaujímavou vlastnosťou týchto článkov sa nepriamo ukázalo aká je dôležitá identifikácia zdroja stránky.

10.4 Zhrnutie

Na základe týchto zistení sa potvrdilo, že pri tvorbe komplexného systému na detekciu falošných správ pre menšie jazykové korpusy je rovnako dôležitá identifikácia falošných správ na základe autora, zdroj a priložených médii ako detekcia pomocou textového obsahu konkrétneho článku. Preto by mal byť tento systém tvorený ako hybridný model, ktorý v sebe spája viacero uhlov pohľadov na falošné správy a hodnotiť ich na základe celkového pohľadu na problematiku fake-news. Výsledné klasifikačné skóre by bolo určené ako kombinácia výsledkov riešení týchto čiastkových problémov. Tento prístup minimalizuje riziko falošne pozitívnej/negatívnej klasifikácie.

11. Základná sada experimentov

V prvej sade experimentov sa zameriame na základné algoritmy strojového učenia, ktoré sme uviedli v kapitole 6. Cieľom týchto experimentov je nastaviť baseline pre komplexnejšie metódy a experimenty pre každý zo slovenských datasetov z kapitoly 9.

Každá podkapitola sa zoberá rovnakou sadou experimentov vykonaných iba na inom slovenskom datasete. Pre všetky dátové sady sú vykonané nasledovné experimenty: V prvom experimente sú porovnané základné metódy strojového učenia pomocou krížovej validácie. V nasledujúcom experimente sme pomocou optimalizácie hyperparametrov našli optimálnu voľbu parametrov pre najlepší klasifikátor z prvého experimentu. Posledný experiment je venovaný Ensemble systémom, u ktorých je predpoklad zlepšenia oproti samostatným klasifikátorom z predošlých experimentov.

11.1 Dataset DownSampled SlovakCovid19 FN

Prvým datasetom, na ktorom sme vykonali základnú sadu experimentov je DownSampled SlovakCovid19 FN. Tento dataset bol nami vytvorený z dôvodu veľkého nevybalansovania pôvodného datasetu, a má slúžiť na vytvorenie základných výsledkov a oboznámenie so slovenskými dátami, ktoré nie sú ovplyvnené nevybalansovanou dátovou sadou. Bližšie informácie o tvorbe tohto datasetu sú uvedené v podkapitole 9.2.2.

Pre tieto experimenty bol dataset predspracovaný ako je uvedené v podkapitole 9.2.2, a v ďalšom kroku rozdelený do troch množín (train, valid, test) v pomere 6:2:2, 60% pôvodných dát tvorí trénovaciu množinu a zvyšných 40% je rozdelených na polovicu medzi validačnú a testovaciu množinu. Rozdelenie prebehlo so zachovaním rozloženia tried v jednotlivých množinách.

11.1.1 Porovnanie algoritmov strojového učenia

Prvým experimentom v základnej sade je porovnanie jednoduchých modelov strojového učenia pomocou krížovej validácie. Pre porovnanie sme si v tomto experimente zvolili nasledujúce algoritmy strojového učenia:

- Rozhodovací strom
- Náhodný les
- k-NN
- Naive Bayesov Klasifikator
- XGB Klasifikátor

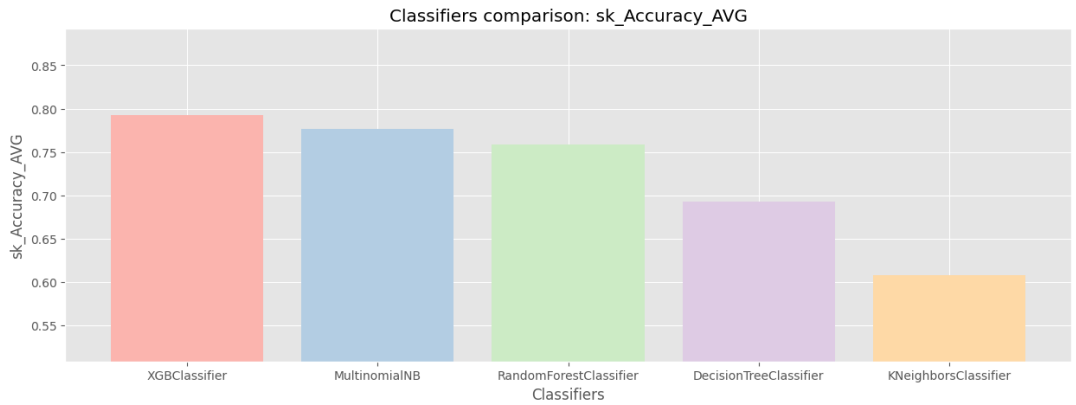
Vyhodnotenie sa dialo pomocou 10-fold krížovej validácie, ktorá slúži na zovšeobecnenie výsledkov pre celú dátovú sadu. Jednotlivé algoritmy strojového učenia sme porovnali na základe nasledujúcich metrík F1 skóre, accuracy a AUC.

Vzhľadom k tomu, že základné algoritmy strojového učenia nie sú schopné pracovať s textovým vstupom, bolo nutné vstupný dataset transformovať do numerickej reprezentácie v procese, ktorý sa nazýva vektorizácia textu. Pre základnú sadu experimentov sme zvolili vektorizáciu textu pomocou algoritmu Bag of words 4.1.1. Parametre vektorizácie textu boli zvolené nasledovne:

- Veľkosť slovnej zásoby 70 000
- Rozmedzie n-gram-ov bolo od 1 - 5
- Nastavené odstránenie slovenských stopwords

Algoritmy strojového učenia, ktoré sme vymenovali vyššie, boli použité v ich základnom nastavení bez špecifickej úpravy ich učiacich parametrov.

V nasledujúcich častiach tejto podkapitoly uvedieme výsledky experimentov a ich interpretáciu. Ako prvé uvedieme vizualizáciu úspešností jednotlivých algoritmov pomocou prehľadných histogramov pre nami zvolené metriky. Na záver všetky výsledky zosumarizujeme v tabuľke, ktorá v sebe bude obsahovať presné hodnoty.

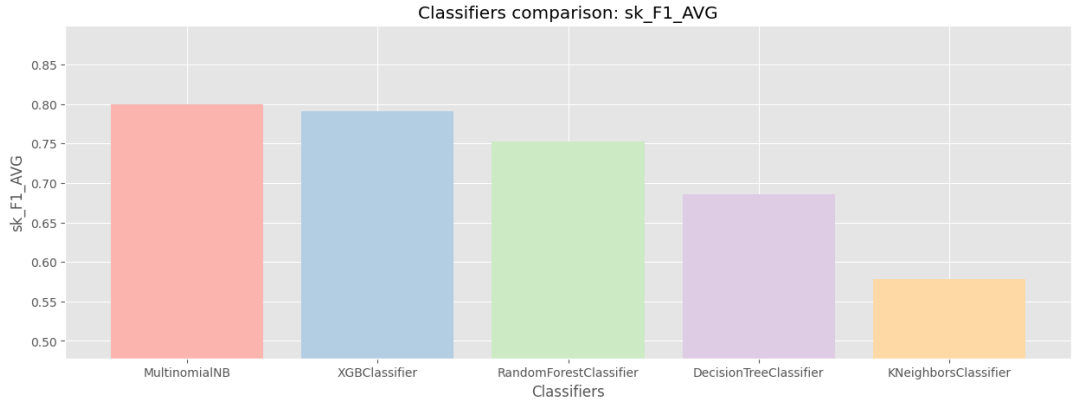


Obr. 11.1: Porovanie klasifikátorov: Accuracy.

Na vizualizácii 11.1 je zobrazený histogram porovnania klasifikátorov na základe priemernej hodnoty metriky accuracy cez všetky behy krížovej validácie jednotlivých algoritmov strojového učenia, ktoré sme testovali.

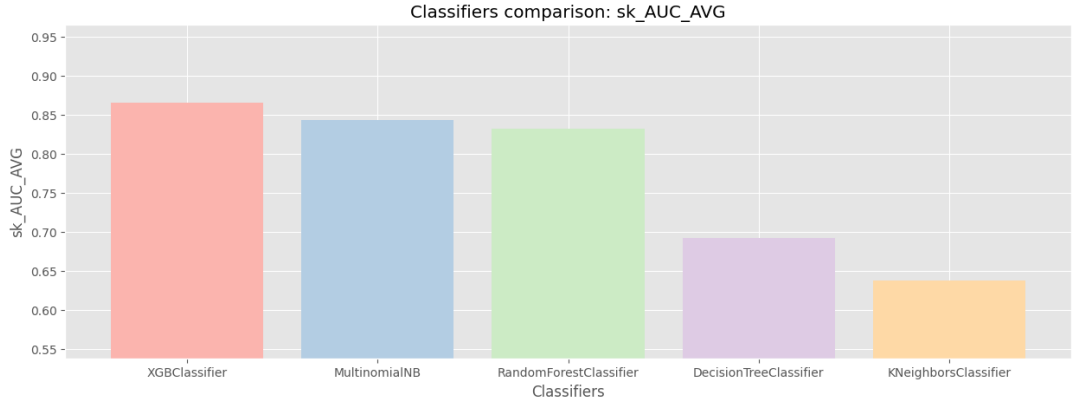
Podľa tejto metriky sa na prvých troch miestach umiestnili nasledovné algoritmy, XGB Klasifikátor, Naive Bayesov Klasifikator a Náhodný les. Prvé dva menované dosiahli podobné skóre, pričom tretí už za prvým značne zaostáva. Nahodný les je horší ako XGB Klasifikátor približne o 4%.

Ďalšie dva algoritmy strojového učenia už nedosahujú kvality prvých troch spomenutých. Posledný dosahuje zhruba accuracy 60% na danom datasete. Na druhú stranu prvý menovaný XGB Klasifikátor dosiahol priemernú hodnotu skóre $acc = 79\%$. Teda najhorší algoritmus dosahuje o 20% nižšiu accuracy ako ten najlepší. Táto metrika ukazuje, že vhodná voľba algoritmu strojového učenia pre riešenie problému detekcie fake-news je zásadná.



Obr. 11.2: Porovanie klasifikátorov: F1 skóre.

Druhá vizualizácia s číslom 11.2 je venovaná porovnaniu algoritmov pomocou metriky F1 skóre. Podobne ako pri predošlej metrike sa na prvých miestach umiestnili Naive Bayesov Klasifikator, XGB Klasifikátor a Náhodný les. Len v tomto prípade si prvé miesto prvé dva algoritmy vymenili. Ale rozdiel v ich úspešnosti je zanedbateľný. Podobnosť medzi vyhodnotením metrík accuracy a F1 skóre je spôsobená vybalansovaným datasetom, teda aj výsledky ostatných algoritmov sú skoro totožné ako pri predošlej metrike.



Obr. 11.3: Porovanie klasifikátorov: AUC skóre.

Posledná metrika na ktorú sa zameriame pri základnej sade experimentov je AUC krivka. Porovnaním jednotlivých algoritmov strojového učenia na základe tejto metriky dostávame poradie, ktoré je totožné so zoradením podľa metriky accuracy. Na obrázku 11.3 je znázornený histogram úspešnosti na základe tejto metriky pre zvolené metódy strojového učenia.

Troma najúspešnejšími algoritmami sú XGB Klasifikátor, Naive Bayesov Klasifikator a Náhodný les. Najúspešnejší algoritmus v tomto prípade je znovu XGB Klasifikátor, ktorý aj v zahraničných štúdiách dosahuje dobré výsledky z pomedzi základných algoritmov strojového učenia pri riešení tohto lingvistického problému, ktorý spočíva v detekcii fake-news.

Najlepšie výsledky, ako môžeme vidieť z grafov zobrazených na ilustráciách 11.1,11.2 a 11.3 dosiahli XGBClassifiar, MultinomialNB a RandomForest.

Na základe týchto vizualizácii sme vo výslednej tabuľke uviedli presné hodnoty iba prvých troch najúspešnejších algoritmov, keďže ďalšie dva algoritmy dosahovali približne o 1/4 horšie skóre podľa nami zvolených metrík, preto ich pre prehľadnosť vo výslednej tabuľke nezobrazujeme.

Klasifikátor	Accuracy	F1 skóre	AUC
XGBClassifier	0.793	0.791	0.866
MultinomialNB	0.777	0.799	0.844
RandomForestClassifier	0.759	0.752	0.832

Tabuľka 11.1: Porovnanie základných klasifikátorov.

Tabuľka 11.1 obsahuje výsledky, v zvolených metrikách, troch najlepších klasifikátorov zo základnej sady algoritmov strojového učenia pre DownSampled SlovakCovid19 FN dataset.

Z výsledkov experimentov v tabuľke 11.1 môžeme vyčítať, že krížová validácia základných algoritmov pre vyvážený a očistený dataset DownSmapled SlovakCovid19 FN dosahuje pomerne dobré výsledky a tieto výsledky môžeme brať ako dobrý odrazový mostík pre ďalšie experimenty.

Aj jednoduché algoritmy vedia s 80% accuracy určiť správnu triedu a to môžeme pokladať pri takto náročnej úlohe klasifikácie fake-news a pri našom obmedzenom množstve vstupných dát za úspech.

Tento výsledok nás ale neprekvapil. Vzhľadom k náročnosti detekcie fake-news boli podobné hodnoty očakávané. Cieľom tohto datasetu bolo oboznámenie sa z datasetom a vytvorenie baseline hranice pre ďalšie experimenty na tejto dátovej sade.

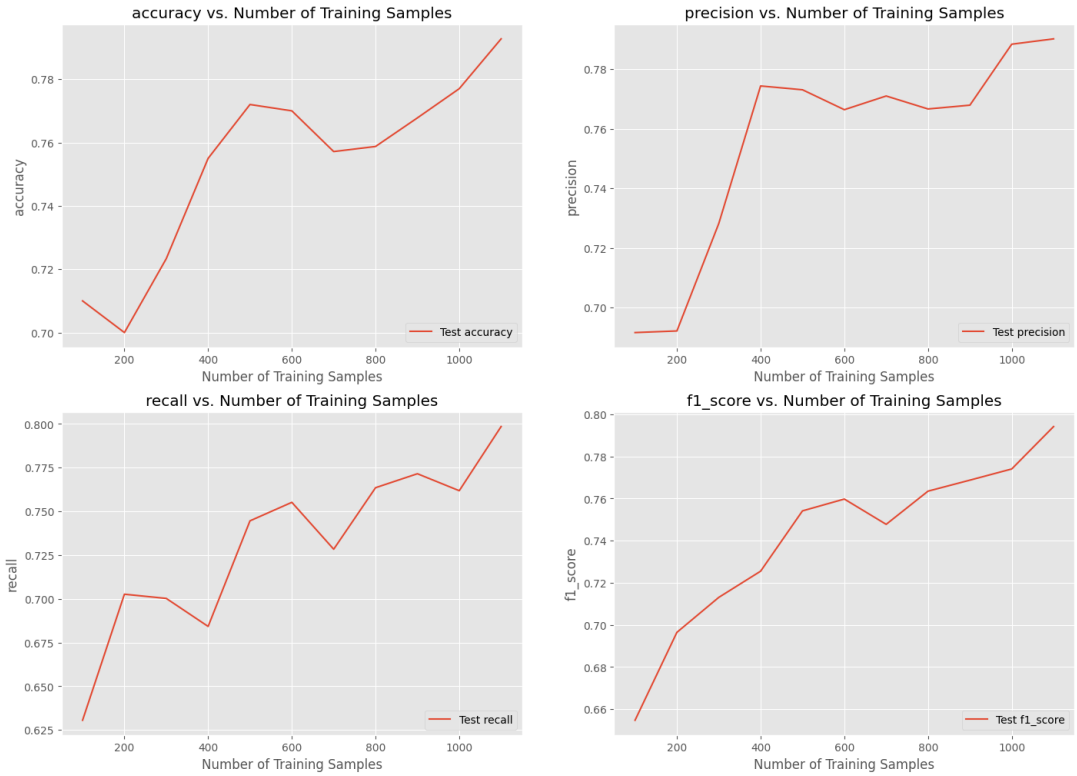
11.1.2 Optimalizácia hyperparametrov

Na základe výsledkov predošlého experimentu sme sa v tomto zamerali na optimalizáciu hyperparametrov pre najlepšie tri metódy z predošlého experimentu. Po optimalizácii hyperparametrov algoritmus XGBClassifier dosahoval najlepšie zlepšenie oproti zvyšným dvom algoritmom. Preto si výsledky optimalizácie tohto algoritmu uvedieme v tejto sekcii.

Rovnako ako v predošlom experimente, základné algoritmy strojového učenia nie sú schopné pracovať s textovým vstupom. Preto bolo nutné vstupný dataset transformovať do numerickej reprezentácie v procese, ktorý sa nazýva vektorizácia textu. Pre hyperoptimalizáciu parametrov sme zvolili nový prístup vektorizácie textu, ktorý dosahoval lepšie výsledky, ktorým je TF-IDF s nasledovnými parametrami:

```
tvec = TfidfVectorizer(
    max_features= 3000,
    ngram_range= (1, 3),
    max_df= 0.65,
    stop_words= stopwords
)
```

Listing 11.1: Hyper. opt.



Obr. 11.4: Vplyv veľkosti trénovacej množiny na skúmané metriky.

Grafy na ilustrácii 11.4 znázorňujú vplyv veľkosti trénovacieho datasetu na výslednú úspešnosť klasifikátora. Zväčšovanie trénovacieho súboru dát ma pozitívny vplyv na vývoj všetkých skúmaných metrík. Síce nastal mierny prepad u všetkých metrík okolo hodnoty 600, ale je iba dočasný a nemá dlhodobý vplyv. Zväčšovanie veľkosti trénovacej množiny pre úlohu klasifikácie fake-news má pozitívny vplyv na výkonnosť modelu.

Výsledkom tohto experimentu teda je, že ak chceme dosiahnuť lepších výsledkov pre klasifikáciu fake-news, je nutné zapracovať na veľkosti trénovacích datasetov, ktoré sú v dnešnej dobre pre slovenský jazykový korpus nedostatočné.

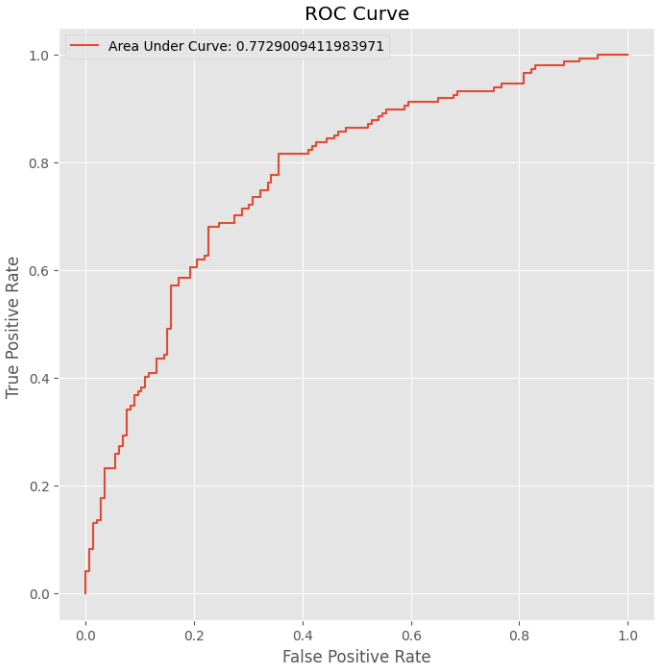
V druhej časti tohto experimentu sme sa zamerali na optimalizáciu hyperparametrov algoritmu XGBClassifier. Na optimalizáciu parametrov sme použili Tree Parzen Estimator. Po 15 iteráciách tohto algoritmu na celom datasete DownSampled SlovakCovid19 FN bola objavená táto množina parametrov ako optimálna:

```
params = {
    'learning_rate': 0.225,
    'max_depth': 5,
    'min_child_weight': 1.0,
    'n_estimators': 600,
    'normalize': 0,
    'scale': 1,
    'subsample': 0.9
}
```

Listing 11.2: Hyper. opt.

Na základe tejto optimalizácie parametrov sme opätovne otestovali XGBClassifier na datasete DownSampled SlovakCovid19-FN s výsledkami $acc = 0.7065$

a $F1 = 0.7062$. Výsledky sú o niečo nižšie ako pri optimalizácii pomocou krížovej validácie, čo je spôsobené vyššou robustnosťou tejto metódy pre menšie dátove sady.



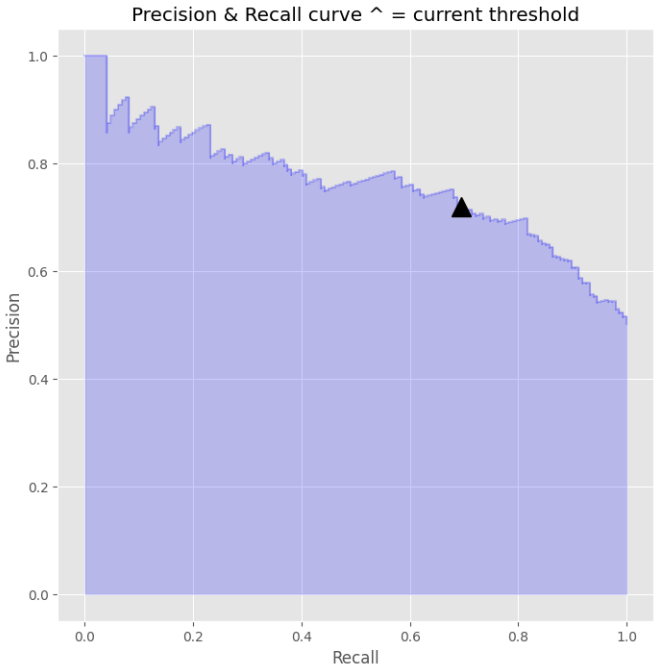
Obr. 11.5: Baseline model: ROC krivka.

Na ilustrácii 11.5 je znázornená ROC krivka, ktorá sa spočítala na základe výsledkov klasifikátora XGB na DownSampled SlovakCovid19 FN datasete. Na základe jeho klasifikácii na testovacej množine sa spočítal TPR a FPR a tie sú vyjadrené v tomto grafe v podobe ROC krivky. Táto krivka nám vyjadruje vzájomný vzťah senzitivity a špecificity daného testu.

Klasifikátor je tým kvalitnejší, čím ma väčšie hodnoty senzitivity a specificity. Ideálna ROC krivka by mala najskôr kolmo stúpať hore a až potom zvyšovať mieru falošnej pozitivity. Ak sa teda pozrieme bližšie na obrázok 11.5, ktorý zobrazuje ROC krivku pre náš baseline klasifikátor XGB, môžeme z neho vidieť, že tento model od ideálneho prípadu má ešte ďaleko. I keď je tam značný logaritmický nárast, ešte stále by mohol byť prudší.

Na druhú stranu náhodný klasifikátor by mal ROC krivku v oblasti diagonály, takže už aj tento jednoduchý klasifikátor, ktorý sme si zvolili za baseline model prekonáva náhodný výber.

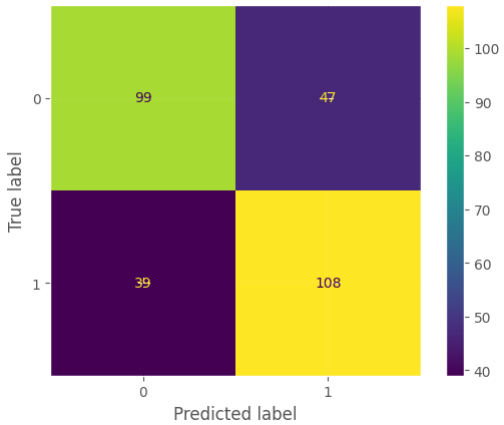
S ROC krivkou súvisí ešte jedná veličina a tou je AUC, ktorá popisuje ROC krivku. A tiež popisuje súčasne kvalitu testu. Nami dosiahnutá hodnota pomocou XGB klasifikátoru bola 0.7729, čo radí kvalitatívne tento klasifikátor ako dobrý model.



Obr. 11.6: Baseline model: Precision recall krivka.

Ďalšou krivkou, ktorá sa používa na vyhodnotenie kvality testovaného klasifikátoru je Precision-Recall krivka. Krivka je vyobrazená pre nami testovaný klasifikátor XGB na obrázku 11.6. Pre túto metriku sme sa rozhodli vzhľadom k porovnaniu s nevybalansovaným datasetom. V tomto experimente, keď sú klasifikované triedy pravdivých článkov a fake-news vyrovnané, nemá o toľko vyššiu výpovednú hodnotu ako ROC krivka. Jej sila sa ukáže až pri nevybalansovanom prípade dát.

Výsledná krivka má očakávaný charakter priebehu. Tento klasifikátor dosahuje mierne nadpriemerné hodnoty ako sme to od baseline modelu očakávali. Na druhú stranu, ešte nedosahuje optimálne výsledky Precision vs Recall.



Obr. 11.7: Baseline model: Konfúzna matica.

Z konfúznej matice XGBClassifier modelu môžme vyčítať, že klasifikuje nesprávne každú štvrtú falošnú správu. A na každé dve správne klasifikovanú falošné správy nesprávne zaradí do tejto množiny aj jeden pravdivý článok.

11.1.3 Ensemble metóda

Posledný experiment, ktorý sme vykonali v základnej sade na DownSampled SlovakCovid119 FN datasete bol zameraný na otestovanie Ensemble metódy a jej prínosu pri klasifikácii fake-news.

Pre tento test Ensemble metód sme si zvolili VotingClassifier, ktorý na základe predošlých experimentov v sebe kombinuje nasledovné základné algoritmy strojového učenia: Náhodný les, Naive Bayesov Klasifikator a XGB Klasifikátor.

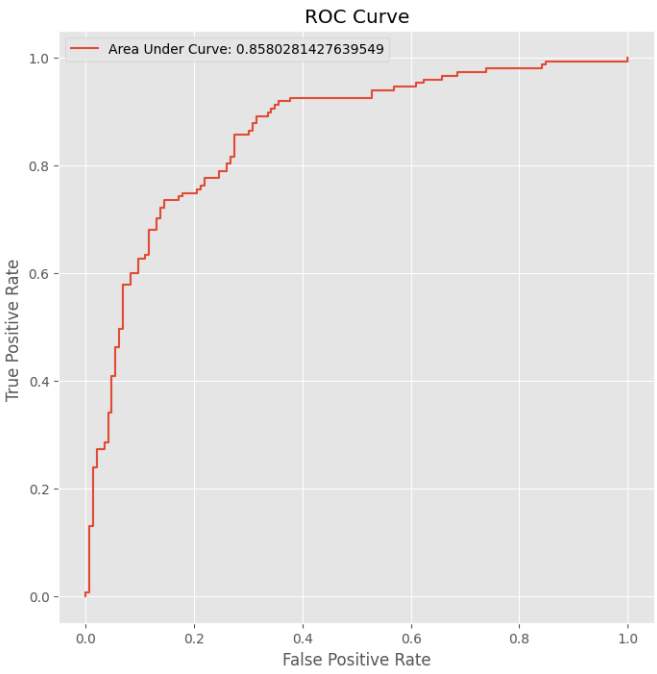
Klasifikátor XGB ma zvolené parametre na základe výsledkov predošlého experimentu hyperoptimalizácie parametrov tohto klasifikátora. Zvyšné dva klasifikátory majú základne nastavenie parametrov.

Keďže aj v tomto teste metód strojového učenia pracujeme s rovnakými algoritmami bola nevyhnutná vektorizácia textov t.j. transformácia textových dát do ich vektorovej podoby. Pre tento experiment sme ďalej pokračovali vo vektorizácii vstupu pomocou TF-IDF algoritmu s týmito parametrami:

```
params = {
    'learning_rate': 0.225,
    'max_depth': 5,
    'min_child_weight': 1.0,
    'n_estimators': 600,
    'normalize': 0,
    'scale': 1,
    'subsample': 0.9
}
```

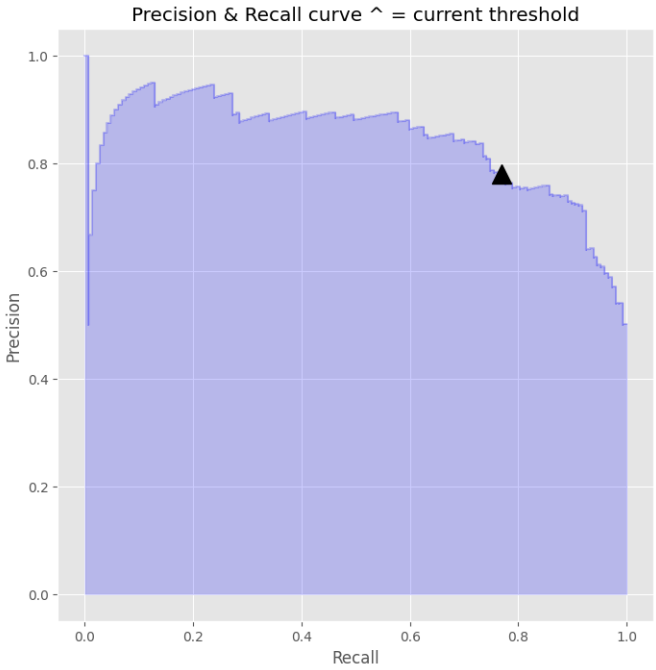
Listing 11.3: Hyper. opt.

Výsledna hodnota klasifikácie je vyjadrená ako priemerná hodnota klasifikácií modelov, ktoré tvoria tento ensemble systém tzv. soft volting. V nasledujúcej časti tejto sekcie popíšeme dosiahnuté výsledky tohto klasifikátoru na DownSampled SlovakCovid19 FN datasete.



Obr. 11.8: Ensemble model: ROC krivka.

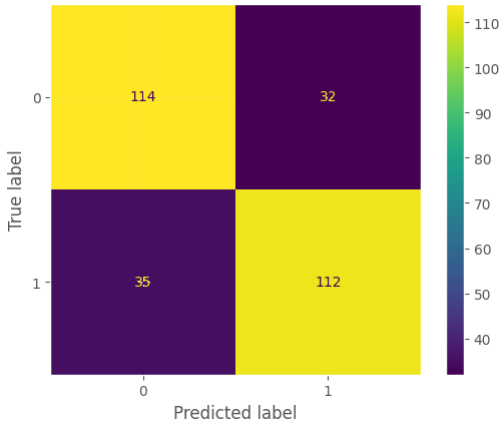
Na grafe 11.8 môžeme vidieť ROC krivku pre testovaný Ensemble systém. Ak priebeh tejto krivky porovnáme s baseline klasifikátorom z predošlého experimentu 11.5 je zjavné, že krivka ensemble systému ma rýchlejší rast, čo interpretujeme ako vyššiu kvalitu modelu, pretože hneď od začiatku nezvyšuje úmerne falošnú pozitivitu. Oproti baseline modelu má bližšie k optimálnemu klasifikátoru na základe ROC krivky. Pre porovnanie s baseline modelom použijeme ešte AUC hodnotu, ktorá súvisí priamo s ROC krivkou. Táto hodnota pre ROC krivku Ensemble systém dosahuje hodnotu 0.8580, kdežto baseline XGB klasifikátor dosiahol hodnotu 0.7729. Ensemble systém sa touto hodnotou dostal už medzi veľmi dobré klasifikátory, na druhú stranu baseline model bol svojim skóre AUC zaradený iba medzi dobré metódy klasifikácie fake-news.



Obr. 11.9: Ensemble model: Precision recall krivka.

Ďalšia metrika na základe, ktorej môžeme Ensemble systém porovnať s baseline modelom je Precision-Recall krivka. Túto metriku môžeme vidieť na vizualizácii 11.9. Rovnako ako pri teste baseline modelu na vybalansovanom datasete nám táto metrika porovnania Precsion vs. Recall veľa nepovie. Ale ak túto krivku porovnáme s Precision recall krivkou pre baseline klasifikátor XGB, môžeme postrehnúť, že pri hodnote precision 0.7 je značný prepad recall až k hodnote 0.4. To môžeme interpretovať ako značné zlepšenie klasifikátoru. Zubatý patern na začiatku krivky sa nazýva „sawtooth pattern“ je pomerne častý pre PR krivky. Tento patern je spôsobený dávkami vzorkových prípadov, ktoré korelujú dobre s výstupnou premennou a vedú k väčšinou správnym pozitívnym výsledkom. Podobne strmé poklesy sa zvyčajne vyskytujú v dávkach vzorkových prípadov, ktoré vedú k väčšinou nesprávnym pozitívnym výsledkom, čo má za následok pokles precision, ale bez dopadu na recall.

Rovnako ako metrika ROC, aj Precision vs Recall potvrdzuje zlepšenie baseline modelu jeho doplnením o ďalšie dva základné modely strojového učenie a ich následne spojenie v ensemble systéme.



Obr. 11.10: Ensemble model: Konfúzna matica.

Na obrázku 11.10 je vizualizácia konfúznej matice Ensemble modelu. Na základe tejto konfúznej matice sme spočítali $acc = 0.7713$ a $F1 = 0.7713$. Výsledné metriky sú približne o 7% vyššie ako pri baseline modely. Toto zlepšenie je markantné a len potvrdzuje výsledky zlepšenia popísane pomocou ROC a Precision vs. Recall kriviek.

Ensemble systém dosahuje lepších výsledkov oproti baseline modelu z dôvodu, že kombinácia viacerých klasifikátorov pridáva robustnosť tomuto modelu a znižuje miss klasifikáciu jednotlivých tried.

11.1.4 Zhrnutie

Cieľom týchto experimentov bolo oboznámenie sa s datasetom, jeho následné spracovanie pomocou metód strojového učenia a vytvorenie Baseline modelu.

V druhom experimente tejto základnej sady sme vytvorili baseline model pomocou XGB klasifikátora, ktorý dosiahol najlepšie výsledky z porovnávaných algoritmov pre DownSampledSlovakCovid-19 FN dataset. Hodnota základných metrík pomocou baseline modelu je stanovená na $acc = 0.7065$ a $F1 = 0.7062$.

Tieto hodnoty budeme pri ďalších experimentoch brať ako referenčné a na základe nich porovnávať prínos komplexnejších modelov pri detekcii fake-news na slovenskom datasete. V poslednom experimente sa nám podarilo vytvoriť systém klasifikátorov zo základných modelov strojového učenia, ktoré prekonali baseline model v každej metrike a tým potvrdili, že nie len zlepšovanie dátovej sady vedie k lepším výsledkom ale aj použitie robustnejších a komplexnejších systémov pre detekciu fake-news.

11.2 Dataset SlovakCovid19 FN

SlovakCovid19 FN je druhým datasetom, na ktorom sme vykonali základnú sadu experimentov. Tento dataset je pôvodným datasetom, ktorý sme získali od Technickej univerzity v Košiciach. Na základe tohto datasetu bola vytvorená downsampled verzia s vybalansovanými množinami tried. Základná sada experimentov pomocou jednoduchých algoritmov strojového učenia bola na zmenšenom datasete predstavená v predošlej podkapitole. Jej úlohou je Slúžiť ako referenčná vzorka pre celý pôvodný dataset.

V tejto podkapitole vykonáme rovnakú sadu experimentov ako na downsampled verzii datasetu a porovnáme ako vplyv zväčšenia dátovej sady, tak aj zmenu rozloženia počtu vzorov v jednotlivých triedach. Tento dataset je značne nevybalansovaný v prospech pravdivých článkov v pomere 93:7. Podobne ako v predošlej sade experimentov bol dataset predspracovaný a rozdelený do troch množín (train, valid, test) v pomere 6:2:2. Touto sadou experimentov chceme ukázať aký vplyv má kvalita a veľkosť datasetu na úspešnosť výslednej klasifikácie.

11.2.1 Porovnanie algoritmov strojového učenia

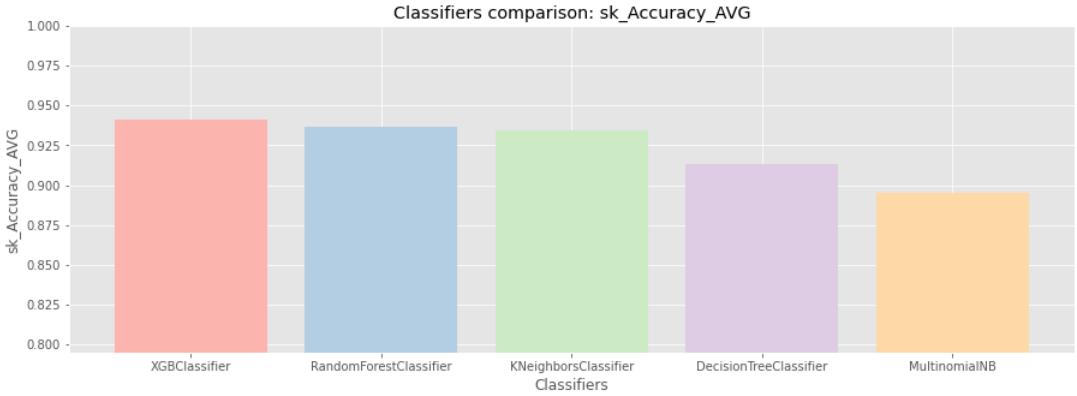
V tomto experimente sme porovnali medzi sebou rovnaké algoritmy strojového učenia ako v podkapitole 11.1. Pre ujasnenie porovnávanými algoritmami sú: Rozhodovací strom, Náhodný les, k-NN, Naive Bayesov Klasifikator a XGB Klasifikátor.

Úspešnosť jednotlivých algoritmov strojového učenia sme pre porovnanie z predošlými výsledkami uskutočnili pomocou rovnakých metrík t.j. F1 skóre, accuracy, AUC. Vyhodnotenie sa dialo pomocou 10-fold krížovej validácie, ktorá slúži na zovšeobecnenie výsledkov pre celú dátovú sadu.

Obdobne ako v predošlých experimentoch základne modely strojového učenia nie sú schopné spracovať s textový vstup aj v tomto prípade bolo nutné previesť transformáciu vstupu v procese vektorizácie. Pre zachovanie korektnosti porovnania experimentov na rôznych dátových sadách aj v tomto prípade prebehla vektorizácia pomocou algoritmu BAG of words s rovnakými parametrami:

- Veľkosť slovnej zásoby 70 000
- Rozmedzie n-gram-ov bolo od 1 do 5
- Nastavené odstránenie slovenských stopwords

Taktiež parametre jednotlivých algoritmov strojového učenia sme ponechali v ich základnom nastavení ako pri predošlom experimente. Najprv ukážeme výsledky experimentov pomocou prehľadných histogramov pre jednotlivé algoritmy a metriky. Nakoniec zhrnieme všetky výsledky v tabuľke s presnými hodnotami.

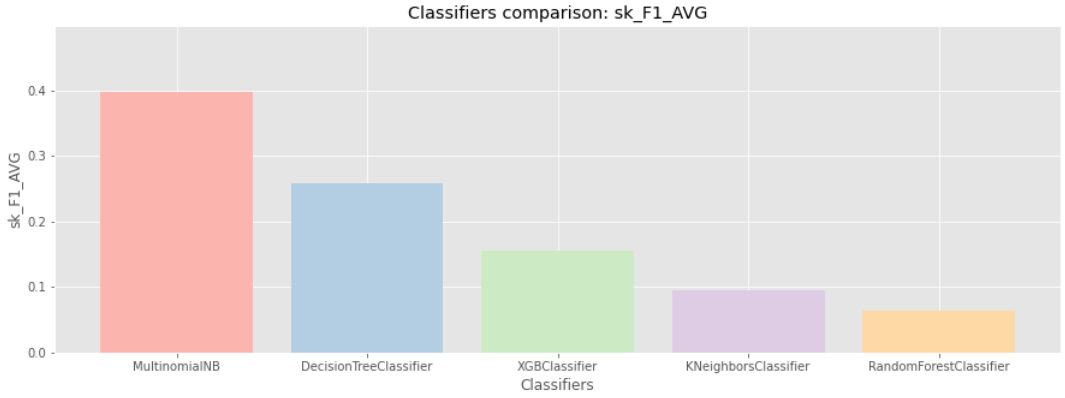


Obr. 11.11: Porovanie klasifikátorov: Accuracy.

Na histograme 11.11 sú porovnané klasifikátory na základe ich priemernej accuracy získanej z krížovej validácie. Ide o výsledky testovania jednotlivých algoritmov strojového učenia na celom slovenskom datasete.

Podľa metriky accuracy cez všetky behy krížovej validácie sa na prvých troch miestach v poradí umiestnili XGB Klasifikátor, Náhodný les a k-NN. Všetky testované algoritmy dosiahli výsledky s rozdielom v presnosti približne 5%. Tento výsledok pre všetky algoritmy bol spôsobený výraznou nerovnováhou rozloženia tried v tomto datasete. A aj keď sa zdá, že na tomto datasete dosahujú jednotlivé algoritmy lepšie výsledky ako na predošlom datasete, nie je tomu tak. Skôr je to naopak a celá skupina testovaných algoritmov dopadla rovnako zle. Tento výsledok nás neprekvapil, pretože dátová sada je značne nevybalansovaná a je ju ťažké klasifikovať.

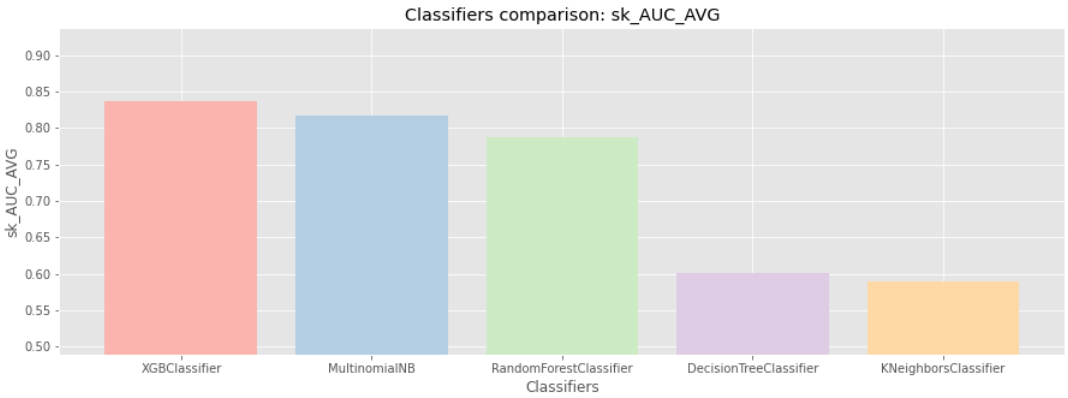
Pre porovnanie, algoritmus ktorý by volil dominujúcu triedu by dosiahol accuracy 0.9381, preto všetky algoritmy, ktoré dosiahli nižšiu hodnotu môžeme pokladať za neúspešné, pretože sú horšie ako tento triviálny spôsob.



Obr. 11.12: Porovanie klasifikátorov: F1 skóre.

Na ilustrácii 11.12 sú výsledky jednotlivých algoritmov na základe metriky F1 skóre. Vizualizáciou tejto metriky pomocou histogramov pre jednotlivé algoritmy sa ukázalo, že pre takto nevyrovnaný dataset je pre porovnanie modelov metrika accuracy nepostačujúca. Ak zohľadníme precision vs recall v metrike F1 skóre. Zistíme, že algoritmy si vedú podpriemerne a dosahujú ani nie 0.5 hodnotu F1 skóre. Aj keď v kontexte nevyváženosti datasetu to nie je zlyhanie je tu vidieť potenciál možného zlepšenia pomocou kvalitnejších modelov.

Pri tejto metrike sa plne ukázal problém celého datasetu a to je nevyváženosť, ktorá bude najväčším problémom aj pri nasledujúcich experimentoch.



Obr. 11.13: Porovanie klasifikátorov: AUC skóre.

V rámci základnej sady experimentov sme sa zamerali aj na metriku AUC krivka. Na obrázku 11.13 je znázornený histogram úspešnosti pre jednotlivé algoritmy podľa tejto metriky. Troma najúspešnejšími algoritmami sú XGB Klasifikátor, Naive Bayesov Klasifikator a Náhodný les. Najúspešnejší algoritmus v tomto prípade je znovu XGB Klasifikátor, ktorý v tejto sade experimentov dokazuje najstabilnejšie výsledky.

Podľa grafov zobrazených na obrázkoch 11.11, 11.12 a 11.13 sú najlepšie výsledky dosiahnuté algoritmami XGBClassifiar, MultinomialNB a RandomForest. Musíme však podotknúť, že najlepšiu stabilitu spomedzi nich mal aj v tomto prípade XGB klasifikátor, ktorý sa vo všetkých metrikách umiestnil v prvej trojici najlepších. Zvyšné dva algoritmy aspoň v jednej z metrík z tejto trojice vypadli.

Základné experimenty ukázali, že najúspešnejšie algoritmy pre detekciu fake-news sú XGBClassifiar, MultinomialNB a RandomForest. Pre prehľadnosť sme vo výslednej tabuľke uviedli iba presné hodnoty prvých troch najúspešnejších algoritmov.

Klasifikátor	Accuracy	F1 skóre	AUC
XGBClassifier	0.941	0.155	0.837
MultinomialNB	0.895	0.394	0.817
RandomForestClassifier	0.937	0.064	0.787

Tabuľka 11.2: Porovnanie základných klasifikátorov.

Ak sa pozrieme na výsledky v tabuľke 11.2 pre prvé tri najúspešnejšie klasifikátory, ktoré boli testované na SlovakCovid19 FN datasete a sú zoradené podľa dosiahnutých hodnôt pre jednotlivé metriky. V tabuľke sú uvedené hodnoty Accuracy, F1 skóre a AUC krivky pre každý z týchto troch klasifikátorov, aby sme mohli porovnať ich výkonnosť a vybrať ten najlepší pre ďalšie experimenty. Ostatné klasifikátory sú z tabuľky vynechané, pretože dosiahli horšie výsledky v porovnaní s týmito tromi najúspešnejšími algoritmami.

Z výsledkov experimentu v tabuľke 11.2 je zjavné, že táto úloha pre celý dataset, ktorý je markantne nevyvážený, je len veľmi ťažko riešitelná. Keďže je dataset nevyvážený v pomere 93:7, úspešnosť v metrike Accuracy menšia ako 0.93 znamená, že aj klasifkátor, ktorý vyberie iba dominantnú triedu je lepší. Túto úspešnosť dosiahli iba 2 klasifikátory, ale tie na druhú stranu, v metrike F1 skóre úplne prepadli a nedosahujú dostatočné výsledky. Znamená to že klasifikujú do triedy pravdivých článkov takmer celý dataset až na pár výnimiek.

Tento výsledok nás je ale očakávaný, vzhľadom k náročnosti detekcie fake-news a tiež vstupným dátam, ktoré sme mali v tomto experimente. Týmto experimentom sme sa chceli oboznámiť s celým pôvodným datasetom a vytvoriť baseline hranice pre ďalšie experimenty, ktoré na výsledkoch týchto klasifikátorov a experimentov budú stavať.

11.2.2 Optimalizácia hyperparametrov

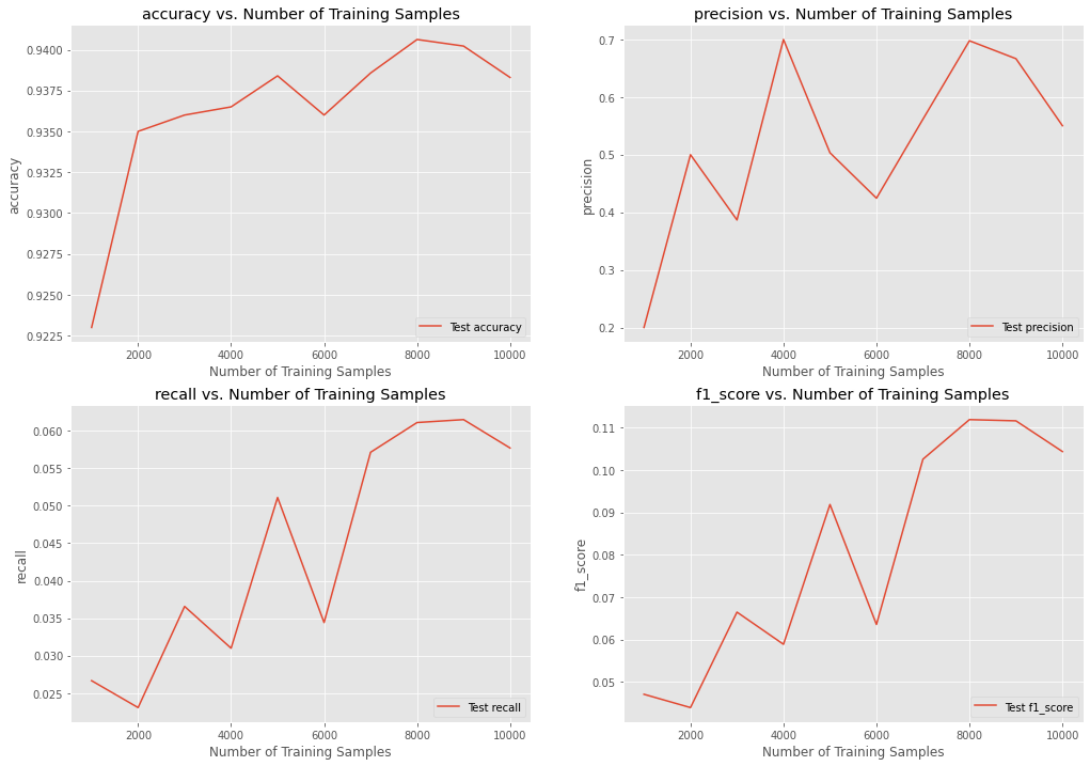
V tomto experimente sme skúmali optimalizáciu hyperparametrov pre najlepšie 3 metódy z predošlého experimentu. V rámci optimalizácie hyperparametrov sme zistili, že algoritmus XGBClassifier má najlepšie výsledky, a to ako v tejto

verzii datasetu, tak aj v downsampled verzii. Preto sa v tejto časti zameriame na výsledky optimalizácie tohto algoritmu.

V tomto experimente sme použili TF-IDF vektorizáciu textu s rovnakými parametrami pre hyperoptimalizáciu ako u downsampled verzie datasetu. Voľba rovnakých parametrov bola z dôvodu korektnosti porovnania optimalizovaných algoritmov.

```
tvec = TfidfVectorizer(
    max_features= 3000,
    ngram_range= (1, 3),
    max_df= 0.65,
    stop_words= stopwords
)
```

Listing 11.4: Hyper. opt.



Obr. 11.14: Vplyv veľkosti tránovacej množiny na skúmané metriky.

Ilustrácia 11.14 obsahuje grafy, ktoré ukazujú, ako veľkosť trénovacieho datasetu ovplyvňuje úspešnosť klasifikátora. Rozšírenie trénovacej množiny vedie k zlepšeniu vývoja všetkých sledovaných metrík. Podobne ako v pri predošlom datasete sme zaznamenali prepad všetkých metrík okolo hodnoty 6000, avšak tento prepad bol dočasný a nemal dlhodobý vplyv. Rovnako tomu bolo aj v predošlej verzii experimentu pre zmenšenú verziu datasetu, tento stav je pravdepodobne spôsobený dodaním novej množiny fake-news, ktorá nebola v doterajších dátach. Značné oscilácie v metrikách sú spôsobené nevyváženou formou datasetu.

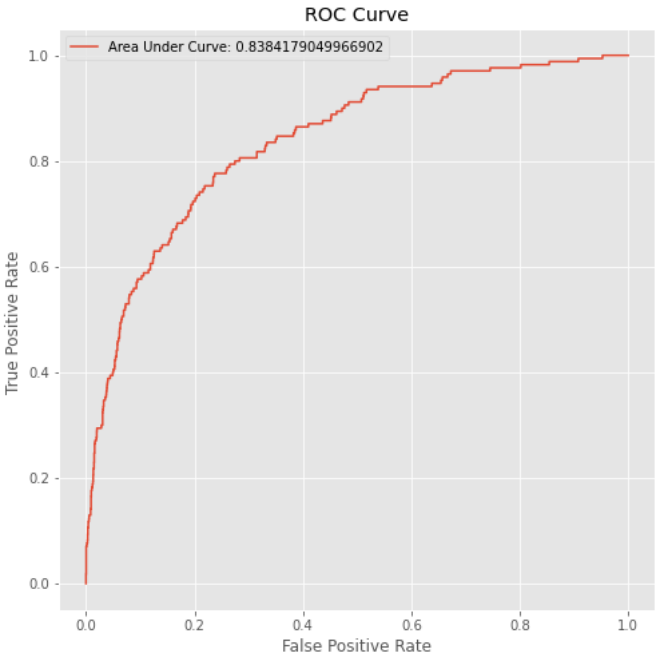
Výsledkom experimentu je, že pre dosiahnutie lepších výsledkov pri klasifikácii fake-news je nutné pracovať na vyváženosti a veľkosti trénovacích datasetov, ktoré pre slovenský jazyk sú momentálne nedostatočné.

Po týchto zisteniach sme sa zamerali na optimalizáciu hyperparametrov algoritmu XGBClassifier. Na optimalizáciu parametrov sme použili Tree Parzen Estimator. Po 15 iteráciách tohto algoritmu na celej trénovacej množine SlovakCovid19 FN sme našli optimálne hodnoty pre hyperparametre modelu, ktoré sú nasledovné:

```
params = {
    'learning_rate': 0.1,
    'max_depth': 5,
    'min_child_weight': 4.0,
    'n_estimators': 150,
    'normalize': 0,
    'scale': 1,
    'subsample': 0.8
}
```

Listing 11.5: Hyper. opt.

Na základe tejto optimalizácie parametrov sme opätovne otestovali XGBClassifier na dátovom súbore SlovakCovid19-FN s výsledkami $acc = 0.9407$ a $F1 = 0.5831$. Po optimalizácii parametrov accuracy modelu neklesla ale F1 skóre zlepšilo svoju hodnotu o 0.4, čo predstavuje zlepšenie o 40% v porovnaní s neoptimalizovanou verziou algoritmu.

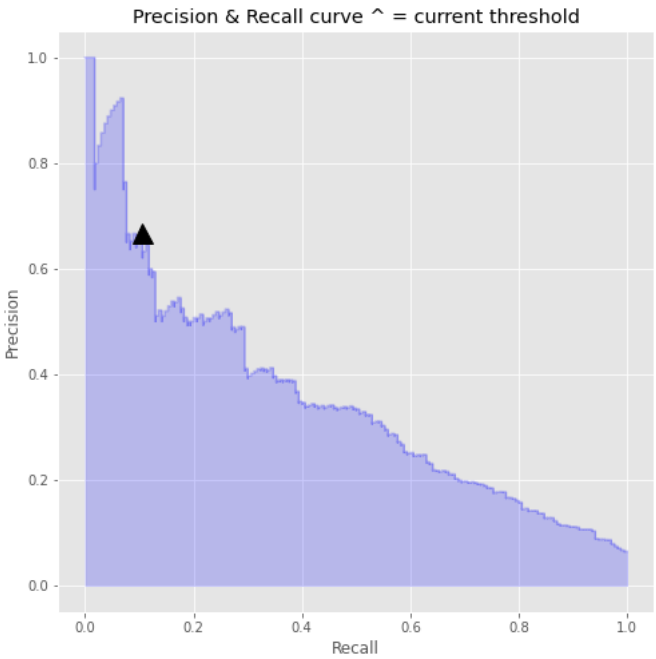


Obr. 11.15: Baseline model: ROC krivka.

Ilustrácia 11.15 zobrazuje ROC krivku, ktorá vyjadruje vzájomný vzťah medzi senzitivitou a specificitou klasifikátora XGB na SlovakCovid19 FN datasete.

Aj keď ROC krivka má pre celý slovenský dataset podobný priebeh ako pre jeho donwsampled verziu musíme tento výsledok brať s rezervou, keďže táto metrika ma značné problémy s nevybalansovanými dátami. AUC je veličina súvisiaca s ROC krivkou, ktorá popisuje kvalitu daného testu. Dosiahnutá hodnota AUC pre klasifikátor XGB na SlovakCovid19 FN datasete bola 0.8384, čo z kvalitatívneho hľadiska radí tento klasifikátor medzi veľmi dobré modely.

Kedže táto hodnota úzko súvisí z AUC krivkou je potrebné brať v úvahu, že hodnoty AUC a ROC krivky môžu byť ovplyvnené nevyváženosťou dát, a preto k ním treba pristupovať s určitou rezervou.



Obr. 11.16: Baseline model: Precision recall krivka.

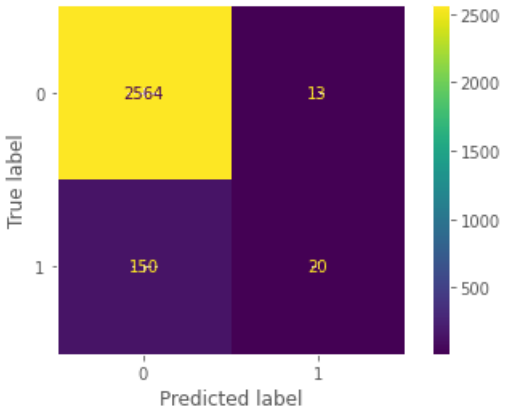
V tejto časti sme použili Precision & Recall krivku na zhodnotenie kvality klasifikátora XGB na SlovakCovid19 FN datasete. Táto krivka bola zvolená z dôvodu nevyvážených tried v datasete. Obrázok 11.16 zobrazuje túto krivku pre náš klasifikátor na zvolenom datasete.

Precision-recall krivka vyjadruje vzťah medzi presnosťou a nájdenými relevantnými dokumentmi pri rôznych hodnotách prahu klasifikácie. Pokles v precíznosti pri recall okolo 0.9 znamená, že pri zvýšení recallu v tomto bode klesá presnosť klasifikátora. Tento jav môže byť spôsobený tým, že v danom bode sa nachádza veľa falošne pozitívnych prípadov, ktoré znižujú presnosť klasifikátora. V praxi to znamená, že ak by sme chceli dosiahnuť vyšší recall, museli by sme akceptovať vyššiu chybovosť klasifikátora v podobe väčšieho počtu falošne pozitívnych výsledkov.

Charakter Precision & Recall krivky pre klasifikátor XGB na našom datasete SlovakCovid19 FN je podobný ako u iných klasifikátorov používaných na podobné úlohy. Krivka sa začína pri hodnote precision blízkej 1, avšak postupne klesá s narastajúcou hodnotou recall. Následne nastáva prudký pokles na hodnotách recall okolo 0.1, kedy sa hodnota precision znižuje z hodnoty 0.9 na približne 0.1.

Tento pokles je spôsobený nevyváženosťou datasetu, kedy máme väčší počet negatívnych príkladov a klasifikátor sa preto snaží minimalizovať počet falošne negatívnych výsledkov. Celkovo však Precision & Recall krivka poskytuje ucelený pohľad na kvalitu klasifikácie v rámci celej rozmanitosti hodnôt thresholdu.

Podľa konfúznej matice klasifikátora XGBClassifier môžeme vidieť, že model správne klasifikuje približne každú siedmu falošnú správu, avšak na každú správne klasifikovanú falošnú správu zaradí nesprávne aj jeden pravdivý článok.

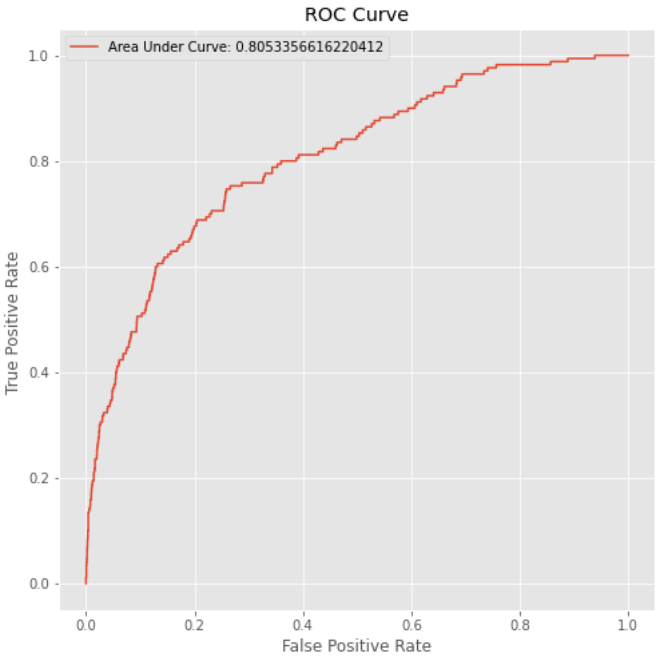


Obr. 11.17: Baseline model: Konfúzna matica.

Tento výsledok bude použitý ako baseline pre verziu celého slovenského datasetu v nasledujúcich experimentoch.

11.2.3 Ensemble metóda

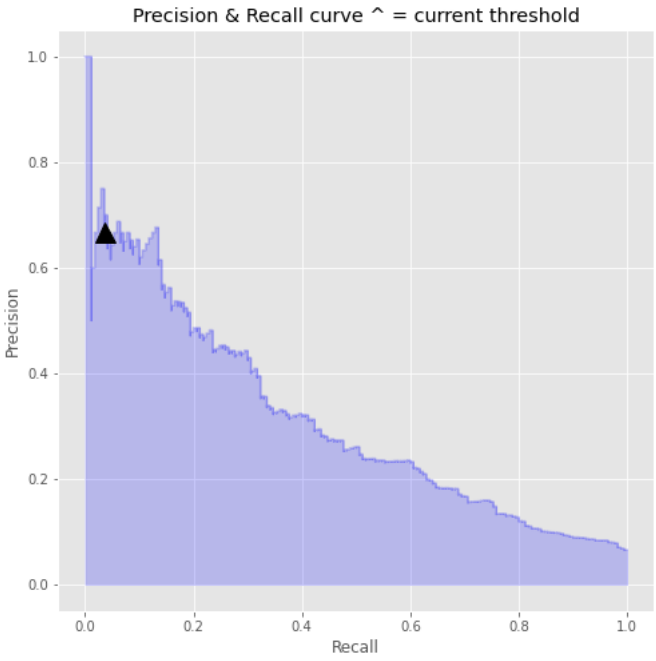
Posledný experiment, ktorý sme vykonali v základnej sade na SlovakCovid19 FN datasete, bol zameraný na otestovanie Ensemble metódy a jej prínosu pri klasifkácii fake-news v nevyváženom datasete. Zvolili sme rovnaký klasifikátor ako pri teste na downsmapled verzii datasetu s rovnakou voľbou parametrov.



Obr. 11.18: Ensemble model: ROC krivka.

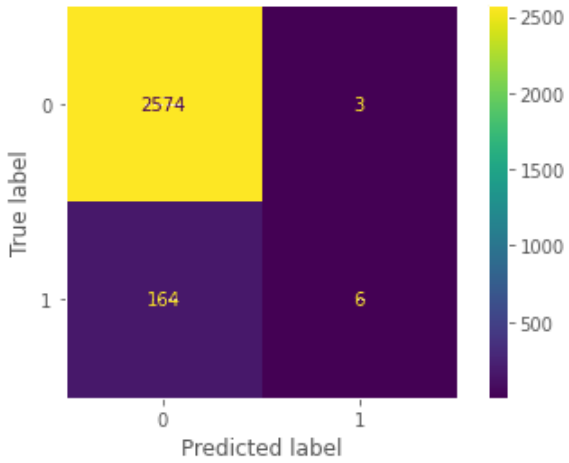
V grafe 11.18 je zobrazená ROC krivka pre testovaný Ensemble systém. Hoci sa na prvý pohľad zdá, že má dobrý charakter priebehu, podobný ako v prípade downsampled datasetu, musíme vziať do úvahy extrémnu nevyváženosť datasetu. Vzhľadom na extrémnu nevyváženosť datasetu nemôžeme považovať ROC krivku

za spoľahlivý ukazovateľ úspešnosti testovaného Ensemble systému. Preto jej hodnoty nezodpovedajú skutočnej úspešnosti systému.



Obr. 11.19: Ensemble model: Precision recall krivka.

Na grafe 11.19 je zobrazená Precision vs. Recall krivka. Z jej prudkého poklesu môžeme vydedukovať, že hneď od samotného začiatku spojením viacerých klasifikátorov pomocou soft metódy výberu predikujeme vyššiu percento falošne pozitívnych prípadov.



Obr. 11.20: Ensemble model: Konfúzna matica.

Táto zvýšená chybovosť systému je spôsobená tým faktom, že dva chybné klasifikácie prehlasovali jednu pozitívnu a indikuje to fakt, že klasifikátory chybne klasifikujú rôzne množiny testovacej množiny a ich samostatná chybovosť sa v tejto skupine ešte znásobuje. V experimente sa potvrdilo, že použitie ensemble metódy, ktorá kombinuje viacero jednoduchých klasifikátorov bez zohľadnenia nevyváženosti dátového setu, vedie k nižšej presnosti modelov v porovnaní s

použitím jednotlivých klasifikátorov samostatne. Tento výsledok bol získaný na datasete s výraznou nevyváženosťou a platí pre tento konkrétny prípad.

Z obrázku 11.20 môžeme vidieť, že kombinovanie viacerých klasifikátorov neprodukuje uspokojivé výsledky. Každý jednoduchý klasifikátor robí chyby v rôznych častiach testovacej sady, ktoré sa prejavujú aj v konečnej klasifikácii. To vedie k tomu, že model predikuje väčšinu príkladov ako pravdivé, ale v skutočnosti má vysokú chybovosť. Tieto výsledky naznačujú, že použitie ensemble modelov nie je vždy vhodné a môže dokonca viesť k zhoršeniu presnosti klasifikácie.

11.2.4 Záver experimentu

Cieľom týchto experimentov bolo oboznámenie sa s celým slovenským datasetom. Jeho následné spracovanie pomocou metód strojového učenia a vytvorenie Baseline modelu.

V druhom experimente tejto sady sme vytvorili baseline pomocou modelu XGBClassifier, ktorý dosiahol najlepšie výsledky z porovnávaných algoritmov. Úspešnosť baseline modelu bola stanovená na $acc = 0.9407$ a $F1 = 0.5831$.

V poslednom experimente tejto sady sme sa rovnako ako pri downsampled verzii snažili vytvoriť ensemble systém pre klasifikáciu fake-news. Musíme však zhodnotiť, že spojením jednoduchých klasifikátorov bez nejakého vyváženia datasetu nemá cenu. Výsledkom tohto experimentu je, že ak chceme na klasifikáciu fake-news použiť jednoduchšie modely je nutné zapracovať na kvalite datasetu. Inak je nutné použiť komplexnejšie metódy strojového učenia.

11.3 Zhrnutie

Za výsledok celej základnej sady experimentov môžeme brať nasledujúce zistenia. Pre kvalitnú detekciu falošných správ je nutné zapracovať na trénovacom datasete, ktorý musí byť viac vybalansovaný a obsahovať väčšie množstvo dát. Skvalitnenie dátovej sady, ako je možné vidieť pri porovnaní dátových sád v tejto kapitole, vedie k veľkému zlepšeniu klasifikácie jednotlivými algoritmami. Na druhú stranu je downsmapled dataset už na hrane svojou veľkosťou, a bolo by vhodné, pri zachovaní vyváženosti jeho tried, ho rozšíriť na väčšiu množinu textov.

Ak bude splnená prvá podmienka skvalitnenia datasetu pre detekciu falošných správ, potom ako ukázali experimenty v tejto sade aj jednoduché algoritmy dokážu detegovať fake-news na slovenských textoch. Na druhú stranu sa ukázalo, že zvýšením komplexity modelov sa táto úspešnosť zvyšuje. Preto sme sa rozhodli v nasledujúcich experimentoch použiť komplexnejšie metódy strojového učenia a porovnať ich na downsampled verzii datasetu. Očakávame pri nich zlepšenie úspešnosti, aj keď za cenu väčšej výpočtovej náročnosti.

12. Experimenty: Neurónové siete

V tejto sade experimentov sa zameriame na úspešnosť hlbokých neurónových sietí, pri klasifikácii falošných správ v slovenskom jazyku, ktoré sme zadefinovali v kapitole 7. Konkrétne typy architektúr neurónových sietí, ktoré boli zvolené pre túto sadu experimentov, boli vybrané na základe ich nadpriemerných výsledkov pri klasifikácii textov. Odôvodnenie sme uviedli v kapitole 7, ktorá pojednáva o neurónových sieťach.

V prvom kroku budeme testovať výkonnosť jednotlivých neurónových sietí na každom zo slovenských datasetov a porovnáme ich úspešnosť na jednotlivých datasetoch. V druhom kroku porovnáme výkonnosť jednotlivých sietí medzi oboma dátovými sadami, aby sme videli, ako dobre sa každá sieť vysporiadala s každým z datasetov.

Tréning neurónových sietí

Všetky neurónové siete boli trénované s týmito parametrami, ak nebude pre daný experiment špecifikované inak.

Parametre: Tréning trval 20/50/100 epoch (pre Downsampled aj hodnota 500) s veľkosťou kroku (batch size) 32, bol použitý optimizer Adam s predvolenou hodnotou learning rate. Metrika behu tréningu bola zvolená presnosť (accuracy) a chybová funkcia Binárna crossentropia (binárny klasifikačný problém).

Evaluácia

Pre každý dataset sme vykonali viacero experimentov s rôznymi nastaveniami parametrov, ktoré sme následne porovnali a vyhodnotili. Výsledky sme prezentovali v prehľadnej tabuľke a pomocou konfúznej matice s podrobným popisom výsledkov. Cieľom týchto experimentov je zistiť, ako dobre sa hlboké neurónové siete dokážu naučiť klasifikovať slovenské textové dáta a aký vplyv majú rôzne parametre na výkonnosť tejto siete.

Rekurentné neurónové siete

Prvá neurónová sieť, ktorú sme zvolili na základe jej nadpriemernej úspešnosti pri klasifikácii textových dát, bola LSTM rekurentná neurónová sieť. Tento typ siete je veľmi populárny pri klasifikácii textových dát s vysokou úspešnosťou, preto sme ju zaradili do tohto experimentu.

V rámci týchto experimentov sme testovali úspešnosť LSTM siete na dvoch slovenských datasetoch pre klasifikáciu falošných správ a porovnávali sme ju s úspešnosťou iných typov neurónových sietí, ktoré sú predstavené v tejto kapitole.

Architektúra siete

Konkrétne sme použili LSTM sieť s nasledujúcou architektúrou, ktorá bola implementovaná v jazyku Python pomocou knižnice *tensorflow*:

```
_________________________________________________________________
 Layer (type)                 Output Shape              Param #
=================================================================
 lstm (LSTM)                  (None, 1024)              16486400
 batch_normalization (BatchN  (None, 1024)              4096
 ormalization)
 dense_0 (Dense)              (None, 512)               524800
 dense_1 (Dense)              (None, 256)               131328
 dense_2 (Dense)              (None, 128)               32896
 dense_3 (Dense)              (None, 64)                8256
 dense_4 (Dense)              (None, 32)                2080
 batch_normalization_1        (None, 32)                128
(BatchNormalization)
 dropout (Dropout)            (None, 32)                0
 dense_5 (Dense)              (None, 1)                 33

=================================================================
Total params: 17,190,017
Trainable params: 17,187,905
Non-trainable params: 2,112
_________________________________________________________________
```

Listing 12.1: LSTM python.

Konvolučná neurónová sieť

Ďalšou neurónovou sieťou, ktorú sme v experimentoch použili, bola konvolučná neurónová sieť. Tento typ siete sa často používa na spracovanie obrazových dát, ale ukázalo sa, že je tiež veľmi úspešný pri klasifikácii textových dát.

V rámci našich experimentov sme taktiež testovali úspešnosť CNN siete pri klasifikácii falošných správ na dvoch slovenských datasetoch a porovnávali sme ju s úspešnosťou LSTM siete a s RCNN sieťou.

Architektúra siete

V rámci tohto experimentu sme použili implementáciu konvolučnej neurónovej siete (CNN) z kapitoly 7 o hlbokých neurónových sieťach, konkrétne z časti 7.4 o CNN. Implementáciu neurónovej siete sme naprogramovali v jazyku Python a použili sme túto architektúru:

```
_________________________________________________________________
 Layer (type)                 Output Shape              Param #
=================================================================
 spatial_dropout1d (SpatialD  (None, 3000, 1)           0
 ropout1D)
 conv1d (Conv1D)              (None, 2996, 128)         768
 dropout_1 (Dropout)          (None, 2996, 128)         0
 conv1d_1 (Conv1D)            (None, 2992, 64)          41024
 dropout_2 (Dropout)          (None, 2992, 64)          0
 max_pooling1d                (None, 1496, 64)          0
(MaxPooling1D)
 flatten (Flatten)            (None, 95744)             0
 dense_1 (Dense)              (None, 512)               49021440
 dense_2 (Dense)              (None, 256)               131328
 dropout_3 (Dropout)          (None, 256)               0
```

```
 dense_3 (Dense)              (None, 128)               32896
 dense_4 (Dense)              (None, 64)                8256
 dense_5 (Dense)              (None, 32)                2080
 batch_normalization (Batc    (None, 32)                128
 hNormalization)
 dropout_4 (Dropout)          (None, 32)                0
 dense_6 (Dense)              (None, 1)                 33

=================================================================
Total params: 49,237,953
Trainable params: 49,237,889
Non-trainable params: 64
_________________________________________________________________
```

Listing 12.2: CNN python.

Rekurentné konvolučné siete

Posledným typom neurónovej siete, ktorý sme použili v tomto experimente je rekurentno-konvolučná neurónovú sieť. Táto sieť kombinuje prvky rekurentnej neurónovej siete a konvolučnej neurónovej siete. Tento typ siete sme bližšie popísali v kapitole 7 o hlbokých neurónových sieťach, konkrétne v sekcii 7.6 o RCNN sieťach, v ktorej je aj porovnanie s predošlými dvoma typmi neurónových sietí pri klasifikácii obrázkov.

Tento typ siete ešte nebol otestovaný radou experimentov, ale my sme sa ju rozhodli použiť, pretože v sebe kombinuje vlastnosti dvoch sietí, ktoré majú veľmi dobré výsledky pri klasifikácii textov. Spolu by teda mohli vytvoriť ešte silnejší systém pre klasifikáciu falošných správ.

Architektúra siete

Podobne ako pri predchádzajúcich dvoch typoch neurónových sietí sme sa pre implementáciu rozhodli použiť jazyk Python a knižnicu *tensorflow*.

```
_________________________________________________________________
 Layer (type)                 Output Shape              Param #
=================================================================
 spatial_dropout1d_1 (Spatia (None, 3000, 1)           0
 lDropout1D)
 conv1d_1 (Conv1D)           (None, 3000, 64)          384
 rcl (RCL)                   (None, 3000, 32)          16000
 dropout_1 (Dropout)         (None, 3000, 32)          0
 rcl_1 (RCL)                 (None, 3000, 32)          12928
 max_pooling1d_1 (MaxPooling (None, 1500, 32)          0
 1D)
 dropout_2 (Dropout)         (None, 1500, 32)          0
 rcl_2 (RCL)                 (None, 1500, 32)          12928
 dropout_3 (Dropout)         (None, 1500, 32)          0
 rcl_3 (RCL)                 (None, 1500, 32)          12928
 max_pooling1d_2 (MaxPooling (None, 94, 32)            0
 1D)
 flatten_1 (Flatten)         (None, 3008)              0
 dense_1 (Dense)             (None, 512)               1540608
 dense_2 (Dense)             (None, 256)               131328
```

```
 dropout_4 (Dropout)          (None, 256)               0
 dense_3 (Dense)              (None, 128)               32896
 dense_4 (Dense)              (None, 64)                8256
 dense_5 (Dense)              (None, 32)                2080
 batch_normalization_1 (Bat   (None, 32)                128
 chNormalization)
 dropout_5 (Dropout)          (None, 32)                0
 dense_7 (Dense)              (None, 1)                 33

=================================================================
Total params: 1,770,497
Trainable params: 1,769,409
Non-trainable params: 1,088
_________________________________________________________________
```

Listing 12.3: RCNN python.

12.1 Dataset DownSampled SlovakCovid19 FN

V tejto časti sme sa zamerali na klasifikáciu slovenských článkov o pandémii COVID-19 pomocou rôznych typov neurónových sietí. Pre tento experiment sme použili Downsampled SlovakCovid19 FN dataset, ktorý sme popísali v kapitole 9 o datasetoch. Tento dataset sme vytvorili zmenšením a vybalansovaním distribúcie klasifikačných tried v pôvodnom slovenskom datasete.

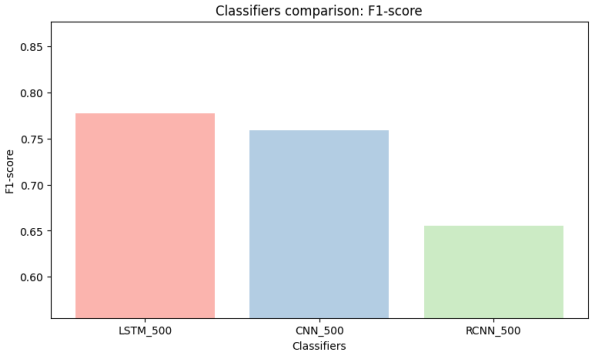
Cieľom tohto experimentu bolo zistiť, ktorý typ neurónovej siete bude mať najvyššiu úspešnosť pri klasifikácii týchto článkov a či bude schopný rozpoznať falošné správy o COVID-19 na slovenských textových dátach.

Experiment	epochs	accuracy	f1-score
LSTM	20	0.7406	0.7369
LSTM	50	0.7235	0.7172
LSTM	100	0.7611	0.7611
LSTM	500	0.7782	0.7771
CNN	20	0.7235	0.7232
CNN	50	0.7474	0.7466
CNN	100	0.7372	0.7371
CNN	500	0.7611	0.7587
RCNN	20	0.4812	0.3565
RCNN	50	0.6485	0.6447
RCNN	100	0.6348	0.6347
RCNN	500	0.6553	0.6553

Tabuľka 12.1: Experiment: Neurónové siete - Downsampled SlovakCovid19 FN.

V tabuľke 12.1 sú uvedené výsledky pre všetky experimenty vykonané pomocou hlbokých neurónových sieti na dátovej sade Downsampled SlovakCovid19 FN. Tabuľka obsahuje hodnoty metrík Accuracy a F1 skóre pre každý model neurónovej siete a počet epoch, ktorých kombináciu sme testovali. Metriky boli zvolené vzhľadom k výsledkom predošlých experimentov, aby boli medzi sebou porovnateľné.

Na obrázku 12.1 je zobrazený histogram, ktorý porovnáva najlepšie modely pre každý typ neurónovej siete s ostatnými metódami v rámci tejto sady experimentov.



Obr. 12.1: Experiment: Neurónové siete - Downsampled SlovakCovid19 FN.

Ako sme uviedli v úvode tejto kapitoly, použili sme tri rôzne typy neurónových sietí: LSTM, CNN a RCNN, s rôznymi hodnotami epoch. Z hodnôt v tabuľke môžeme vydedukovať, že najlepšie výsledky pri klasifikácii falošných správ o COVID-19 na slovenských textových dátach sme dosiahli s LSTM sieťou s najväčším počtom epoch t.j. rovný 500. Táto sieť dosiahla accuracy 0.7782 a f1-score 0.7771. Tesne za touto sieťou nasleduje CNN, rovnako s 500 epochami, ktorej úspešnosť bola, accuracy 0.7611 a f1-score 0.7587. RCNN si viedla o poznanie horšie. Najlepší výsledok podobne ako predošlé dve siete dosiahla s maximálnym počtom epoch. Avšak, jej výsledná úspešnosť bola v porovnaní s ostanými neurónovými sieťami, ktoré sme v tomto experimente testovali o zhruba 10% nižšia pre obe sledované metriky. Úspešnosť RCNN bola nasledovná accuracy 0.6553 a f1-score 0.6553. Úspešnosť jednotlivých sieti je možno vizuálne porovnať na obrázku ilustrácii 12.1, na ktorej je zobrazený histogram metriky f1-skóre.

Všeobecne môžeme vidieť, že pre každý typ siete sme dosiahli najlepšie výsledky s väčším počtom epoch. Pri testovaní vyšších počtov iterácii neurónové siete začali trpieť problémom preučenia a ich výkonnosť klesla. LSTM a CNN siete sa javia z hľadiska úspešnosti ako lepšie v porovnaní s RCNN sieťou. Na druhú stranu v prípade RCNN došlo k preučeniu až s vyšší počtom epoch, ktorý sa však iba vyrovnal predošlým dvom neurónovým sieťam.

Ak porovnáme neurónové siete s baseline model, ktorý dosiahol tieto výsledky $acc = 0.7065$ a $F1 = 0.7062$ na tejto dátovej sade. Dvojica sieti LSTM a CNN, tento model prekonala o 7%, čo môžeme brať ako ohromné zlepšenie v klasifikácii fake-news. Toto zlepšenie predstavuje v detekcii veľkú zmenu oproti baseline modelu. Konkrétne došlo k zlepšeniu, kedy baseline modelu zo štyroch správ unikne jedna na hodnotu úniku jednaj správy zo šiestich. RCNN ako jediná neurónová sieť tento model neprekonala počas 500 epoch, ktoré sú uvedné vo výsledkoch. Aby dosiahla porovnateľnú úspešnosť s ostatnými neurónovými sieťami, musela byť táto sieť trénovaná s väčším počtom epoch. Aj keď to bolo výpočtovo náročné, stále nedokázala prekonať ostatné neurónové siete v presnosti klasifikácie. Tento fakt predpokladáme, je spôsobený komplexnejšou konštrukciou tohto modelu, ktorý je kombináciou predošlých dvoch architektúr. Tento model k natrenovanie váh potrebuje rozsiahlejší dataset alebo vyšší počet epoch, čo sa aj ukázalo pri jeho zlepšovaním pomocou počtu iterácii, ktoré sú uvedené v tabuľke 12.1.

Z tohto experimentu si odnášame nasledovné zistenia: Komplexnejšie modely strojového učenia dosahujú už v svojom základnom nastavení lepšie výsledky ako optimalizované klasifikátory z predošlých testov. Predpokladáme, že ďalší výskum a optimalizácia týchto modelov by mala viesť ešte k zvýšeniu úspešnosti neurónových sietí. Vzhľadom k tomu, že už tieto základne NN prekonali baseline model sa posunieme v tejto práci v ďalších experimentoch k ešte komplexnejším modelom typu Transformer, u ktorých predpokladáme ešte markantnejšie zlepšenie úspešnosti. Na druhú stranu z tohto experimentu možeme vyčítať, že slovenský dataset v takejto kvalite (vybalansovaný) nie je dostatočne rozsiahly a je potrebné jeho rozširovanie a zlepšovanie, aby mohla byť realizovaná automatická detekcia falošných správ na slovenských dátach.

12.2 Dataset SlovakCovid19 FN

Rovnako ako v predošlom experimente aj v tomto sme sa zamerali na klasifikáciu falošných správ pomocou rôznych typov neurónových sietí na slovenských textových dátach. Pre tento experiment sme použili celý pôvodný slovenský dataset pre klasifikáciu fake-news.

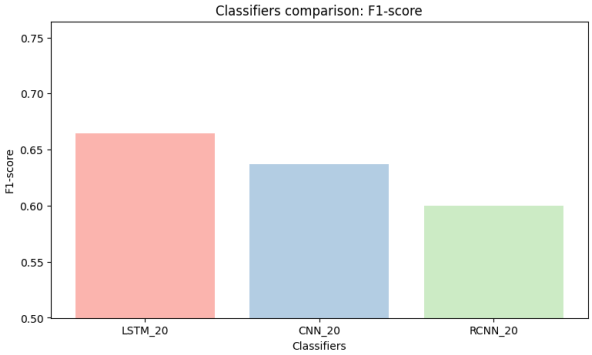
Cieľom tohto experimentu bolo rovnako porovnať úspešnosť rôznych typov neurónových sietí pri klasifikácii falošných správ na rozsiahlejších slovenských textových dátach a zistiť, ktorý typ siete dosahuje najlepšie výsledky. Druhým cieľom bolo porovnať vplyv kvality datasetu na úspešnosť neurónových sietí. Na to nám poslúžia výsledky z predošlého experimentu, ktorý bol vykonaný na vybalansovanej verzii tohto datasetu.

Experiment	epochs	accuracy	f1-score
LSTM	20	0.9370	0.6643
LSTM	50	0.9356	0.6248
LSTM	100	0.9327	0.6359
CNN	20	0.9388	0.6369
CNN	50	0.9327	0.5713
CNN	100	0.9403	0.6151
RCNN	20	0.8897	0.5998
RCNN	50	0.9276	0.5877
RCNN	100	0.9301	0.5454

Tabuľka 12.2: Experiment: Neurónové siete - SlovakCovid19 FN.

Vzhľadom na to, že sme vykonali podobný experiment na zmenšenom datasete, uvádzame výsledky v rovnakom formáte ako v predchádzajúcej časti. V tabuľke 12.2 sú uvedené výsledky pre všetky experimenty s hlbokými neurónovými sieťami na SlovakCovid19 FN datasete. Podobne na ilustrácii 12.2 je histogram f1 skóre pre najlepšie modely, ktoré sme testovali na dátovej sade SlovakCovid19 FN.

Z týchto výsledkov by sa mohlo na prvý pohľad zdať, že všetky testované modely dosiahli vysokú presnosť klasifikácie, ale v skutočnosti to skresľuje nevybalansovanosť datasetu. Úspešnosť $acc = 0.938$ dosahuje aj klasifikátor, ktorý volí dominantnú triedu. Ak sa pozrieme na výsledky už s touto informáciou, po-



Obr. 12.2: Experiment: Neurónové siete - SlovakCovid19 FN.

tom dostatočne kvalitný výsledok dosahuje iba jeden typ neurónovej siete a tou je CNN. Na druhú stranu, ak sa pozrieme na F1 skóre, LSTM sieť dosahuje lepších výsledkov. To znamená, že detekuje viac falošných správ, ale za cenu vyššej falošnej pozitivity. V porovnaní s predchádzajúcimi experimentmi na zmenšenom datasete sa ukázalo, že neurónové siete majú v základnom nastavení mierne lepšie výsledky ako baseline. Avšak rozdiel už nie je tak výrazný ako v predchádzajúcich experimentoch. Porovnanie výsledkov z tejto sady testov ukazuje, že komplexnejšie strojové učenie prináša zlepšenia, hoci nie tak značné ako v prípade downsampled verzie datasetu. Tento výsledok potvrdzuje hypotézu, že zlepšovanie klasifikácie falošných správ závisí na zlepšovaní modelov aj datasetu. Preto sme sa rozhodli vytvoriť DownSampled SlovakCovid19 FN dataset, ktorý sme predstavili v predošlých experimentoch a v kapitole 9.

12.3 Zhrnutie

V tejto sade experimentov sme implementovali 3 rôzne architektúry neurónových sietí z kapitoly 7. Ich výkonnosť sme porovnali na klasifikačnej úlohe detekcie fake-news na pôvodnom datasete SlovakCovid19 FN.

Za výsledky tejto sady experimentov považujeme potvrdenie hypotéz, ktoré sú zo zahraničných štúdií ale aj z našich predošlých experimentov. Použitie komplexnejších t.j. zložitejších metód strojového učenia, vedie k zlepšeniu úspešnosti predikcie falošných správ. Tento poznatok a zistenie sme predpokladali, preto v ďalších experimentoch budeme pokračovať v tomto trende a použijeme ešte komplexnejšie metódy detekcie fake-news. Na druhú stranu sa v tomto experimente ukázala ešte viac dôležitosť kvalitných dát oproti základnej sade testov. Tento test zdôraznil dôležitosť kvalitných dát pre ich využitie pre automatickú detekciu fake-news v slovenskom jazyku. Ukázali sme, že slovenský dataset v takejto kvalite nie je dostatočný a je potrebné jeho vybalansovanie a rozširovanie, ak chceme pokročiť v riešení tejto problematiky. Vybalansovanie nie je ten najdôležitejší problém, ale spolu s malým množstvom falošných správ spôsobuje problémy pri učení klasifikátorov. Preto je potrebné vybudovanie kvalitnejšej dátovej sady, aby sa následne zlepšila automatická detekcia falošných správ.

13. Experimenty: Transformers

V tomto experimente porovnáme modely založené na architektúre Transformer pri klasifikácii falošných správ v slovenskom jazyku, ktoré sme zadefinovali v kapitole 8.

V prvom kroku budeme testovať výkonnosť rôznych architektúr typu Transformer na každom zo slovenských datasetov a porovnáme ich úspešnosť.

V druhom kroku porovnáme výkonnosť jednotlivých modelov po ich finetuningu na anglických datasetoch zameraných na falošné správy. Evaluácia však vždy prebehne na rovnakých slovenských dátach ako v predošlých experimentoch.

V poslednom experimente sa zameráme na spojenie slovenského trénovacieho datasetu a anglických datasetov. Pri tomto experimente najskôr prebehne finetuning na anglickom datasete a v ďalšom kroku na slovenskom trénovacom datasete. Evaluácia rovnako ako v predošlom prípade prebehne na testovacom slovenskom súbore dát.

Výsledky jednotlivých experimentov uvedieme v prehľadnej tabuľke a v histograme najúspšnejších modelov pre architektúru Transformer.

Tokenizácia vstupných textov prebehla s nasledovným nastavením parametrov:

```
# Tokenizer parametre
    add_special_tokens = True
    max_length = 512
    pad_to_max_length = True
    return_attention_mask = True
```

Listing 13.1: Tokenizer: parametre

Parametre fine-tuningu sú pre jednotlivé modely nastavené na rovnakú hodnotu za účelom porovnania výkonnosti týchto modelov medzi sebou.

```
# Model parametre
  batch_size=8
  epochs=20
  loss = SparseCategoricalCrossentropy(from_logits=True)
  metric = SparseCategoricalAccuracy('accuracy')
  optimizer = Adam(learning_rate=2e-5,epsilon=1e-08)
```

Listing 13.2: Model: parametre

Všetky modely v tejto sade experimentov sú implementované v jazyku Python pomocou knižnice *tensorflow*. Konkrétne typy architektúr Transfomer, ktoré boli zvolené pre túto sadu experimentov sú nasledovné:

BERT

Ako prvý typ architektúry Transformer sme si zvolili model BERT. Voľba na tento typ architektúry padla pre jeho popularitu medzi NLP komunitou, ale hlavne z dôvodu experimentu založeného na prenositeľnosť znakov falošných správ.

V rámci testov je zvolená anglická verzia tohto modelu, preto je nutné slovenské dátové sady pre tento model automaticky preložiť do anglického jazyka.

Týmto krokom otestujeme prenositeľnosť poznávacích znakov fake-news medzi jazykmi.

Architektúra

Konkrétne bola zvolená base verzia architektúry modelu BERT. Tento typ architektúry transformer sme tiež zvolili z dôvodu porovnanie s modelom Slovak-BERT, ktorý ma rovnaký počet parametrov.

```
_________________________________________________________________
 Layer (type)                 Output Shape              Param #
=================================================================
 bert (TFBertMainLayer)      multiple                  109482240

 dropout_37 (Dropout)        multiple                  0

 classifier (Dense)          multiple                  1538

=================================================================
Total params: 109,483,778
Trainable params: 109,483,778
Non-trainable params: 0
_________________________________________________________________
```

Listing 13.3: BERT-base-uncased

mBERT

Ďalším modelom v našej sade je multilingválny BERT, ktorý sme vybrali z dôvodu, možnosti vyhnúť sa nutnosti prekladu textov pri použití predténovaných cudzojazyčných modelov s architektúrou Transformer.

V rámci testovania tohto modelu budeme pracovať s textami, ktoré sú preložené, a aj tie, ktoré sú v pôvodnej forme. Je tiež dôležité poznamenať, že multilingválny BERT disponuje širšou škálou jazykových zdrojov v porovnaní s jednojazyčnými modelmi. To nám umožní pracovať s textami v rôznych jazykoch.

Architektúra

Podobne ako v predošlom prípade modelu BERT sme zvolili base verziu, z dôvodu korektnosti porovnania so slovenskou verziou modelu BERT.

```
_________________________________________________________________
 Layer (type)                 Output Shape              Param #
=================================================================
 bert (TFBertMainLayer)      multiple                  109482240

 dropout_37 (Dropout)        multiple                  0

 classifier (Dense)          multiple                  1538

=================================================================
Total params: 109,483,778
Trainable params: 109,483,778
Non-trainable params: 0
```

```
14 ----------------------------------------------------------------
```

Listing 13.4: mBERT-base-uncased

RoBERTa

RoBERTa je ďalšou verziou modelu založeného na architektúre Transformer. Tento model vznikol na základe rozšírenia pôvodnej verzie BERT o nové trénovacie techniky, ktoré umožnili vylepšiť jeho výkon v rôznych jazykoch. RoBERTa dosahuje lepšie výsledky v porovnaní s BERT-om a môže byť použitý na širokú škálu jazykových úloh.

Architektúra

RoBERTa-base je založená na rovnakej architektúre ako BERT-base. Avšak v porovnaní s BERT-om má RoBERTa-base viacero trénovacích parametrov, čím sa dosahuje lepší výkon v rôznych jazykoch.

RoBERTa-base má viac než 120 miliónov trénovacích parametrov a dosahuje výrazne lepšie výsledky v rôznych jazykových úlohách ako BERT-base. Táto verzia RoBERTa bola trénovaná na viac ako 160 GB textových dát v rôznych jazykoch a dosahuje lepší výkon aj v prípade, že trénovacie dáta sú nevyvážené.

```
1  ----------------------------------------------------------------
2   Layer (type)                 Output Shape              Param #
3  ================================================================
4   roberta (TFRobertaMainLayer  multiple                  124055040
5   )
6
7   classifier (TFRobertaClassi  multiple                  592130
8   ficationHead)
9
10 ================================================================
11 Total params: 124,647,170
12 Trainable params: 124,647,170
13 Non-trainable params: 0
14 ----------------------------------------------------------------
```

Listing 13.5: RoBERTa- base

XLM-RoBERTa

XLM-RoBERTa je založená rovnako ako predošlé modely na architektúre Transformer model, ale s niektorými vylepšeniami, ktoré umožňujú prekladať text medzi rôznymi jazykmi. XLM-RoBERTa je teda rozšírením RoBERTa o možnosť práce s viacerými jazykmi.

XLM-RoBERTa má viac ako 277 miliónov trénovacích parametrov, čo je oveľa viac ako RoBERTa-base. Táto verzia bola trénovaná na ešte väčšom objeme textových dát z viacerých jazykov, ako predošlé modely. Preto je schopná dosiahnuť ešte lepšie výsledky v multilingválnych úlohách ako mBERT.

Architektúra

Architektúra modelu XLM-RoBERTa implementovaná v jazyku Python pomocou knižnice *tensorflow*.

```
_________________________________________________________________
 Layer (type)                 Output Shape              Param #
=================================================================
 roberta (TFXLMRobertaMainLa  multiple                  277453056
 yer)

 classifier (TFXLMRobertaCla  multiple                  592130
 ssificationHead)

=================================================================
Total params: 278,045,186
Trainable params: 278,045,186
Non-trainable params: 0
_________________________________________________________________
```

Listing 13.6: XLM-RoBERTa-base

SlovakBERT

SlovakBERT je posledným modelom v tejto sade experimentov, ktorý je založený na architektúre Transformer. Tento model má potenciál výrazne zjednodušiť a zrýchliť prácu s jazykovými dátami v slovenskom jazyku a pomôcť tým pádom aj vývoju jazykových technológií v Slovenskej republike.

Výhľadom k tomuto potenciálu sme sa ho rozhodli zaradiť do tejto sady experimentov a porovnať jeho úspešnosť s cudzojazyčnými verziami prípadne multilingválnymi. Cieľom je otestovať, či modely pre konkrétne jazyky aj v dnešnej dobe strojového prekladu poskytujú konkurenčnú výhodu pri spracovaní prirodzeného jazyka. Konkrétne v našom prípade pri detekcii falošných správ v slovenskom jazyku.

Architektúra

Architektúra SlovakBERT modelu je rovnako ako v predošlých modeloch rozšírená iba o klasifikačnú vrstvu.

```
_________________________________________________________________
 Layer (type)                 Output Shape              Param #
=================================================================
 bert (TFBertMainLayer)       multiple                  109482240

 dropout_37 (Dropout)         multiple                  0

 classifier (Dense)           multiple                  1538

=================================================================
Total params: 109,483,778
Trainable params: 109,483,778
Non-trainable params: 0
_________________________________________________________________
```

Listing 13.7: SlovakBERT

13.1 DownSampled SlovakCovid19 FN

Fine-tuning modelov a ich evaluácia prebehne na DownSampled SlovakCovid19 FN datasete, ktorý sme vytvorili ako súčasť tejto práce. Nami vytvorený dataset v tomto experimente rozdelíme v pomere 6:2:2 na trénovaciu, validačnú a testovaciu množinu.

Následne pre anglický model BERT je nutné každú z týchto množín preložiť do anglického jazyka na čo sme využili automatický preklad v Pythone pomocou knižnice *googletrans*. Model mBERT je multilingválna verzia BERT-u, ktorú v tomto experimente otestujeme ako na pôvodných slovenských dátach tak aj na preloženej verzii.

Ďalšia dva modely založené na architektúre Transformer, ktoré sme si predstavili v kapitole 8 a ich implementáciu sme uviedli v úvode tejto kapitoly sú RoBERTa a XLM-RoBERTa. Podobne ako pri predošlých dvoch zahraničných modeloch ich otestujeme na preloženej verzii slovenského downsampled datasetu. V prípade multilingválnej verzie RoBERTa na oboch variantách ako pri mBERT.

Posledným modelom, ktorý sa zúčastni tohto experimentu je prvý slovenský model založený na architektúre Transformer, SlovakBERT.

Pre multilingválny BERT/RoBERTa použijeme fine-tuning v dvoch samostatných verziách. Označme prvú verziu mBERT/RoBERTa, ktorá bude prispôsobená na datasete v slovenskom jazyku. Potom mBERT*/RoBERTa* je verzia modelu prispôsobená na preloženom slovenskom datasete do anglického jazyka. O automatický preklad do angličtiny sa postará Python knižnica *googletrans*.

Výsledky

Výsledky pre modely založené na architektúre Trasformer sú uvedené v tabuľke 13.1 a na ilustrácii 13.1, na ktorej sú jednotlivé modely zoradené podľa metriky F1 skóre. Hviezdička pri multilingválnych modeloch mBERT*/RoBERTa značí, že tento experiment bol vykonaný na preloženom slovenskom datasete.

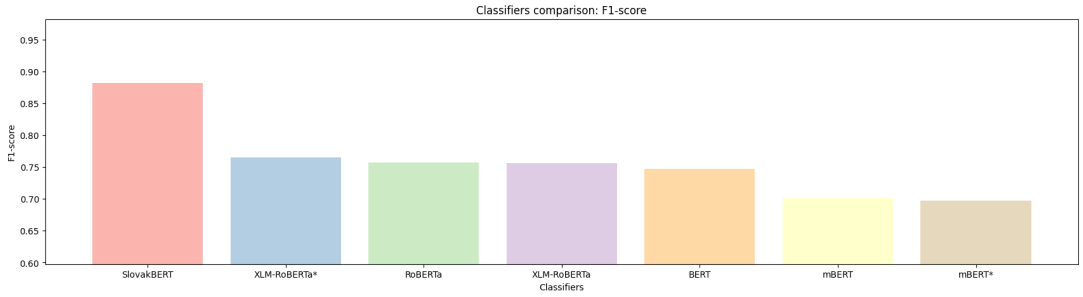
Experiment	epochs	precision	recall	accuracy	f1-score
BERT	20	0.7836	0.7143	0.7577	0.7473
mBERT*	20	0.8108	0.6122	0.7338	0.6977
mBERT	20	0.7661	0.6463	0.7235	0.7011
RoBERTa	20	0.7730	0.7415	0.7611	0.7569
XLM-RoBERTa*	20	0.7899	0.7415	0.7713	0.7649
XLM-RoBERTa	20	0.7868	0.7279	0.7645	0.7562
SlovakBERT	20	0.8491	0.9184	0.8771	0.8824

Tabuľka 13.1: Experiment: Transformer Downsampled SlovakCovid19 FN.

Výsledky v tabuľke 13.1 pre jednotlivé modely sú prezentované pomocou metrík precision, recall a F1-score, ktoré sú bežne používané pri hodnotení výkonu klasifikačných modelov.

Z tabuľky vyplýva, že najlepším modelom pre tento dataset je SlovakBERT, ktorý dosiahol najvyššie hodnoty pre všetky sledované metriky. Tento model dosiahol precision 0.8491, recall 0.9184 a f1-score 0.8824. Na druhom mieste sa

umiestnil model XLM-RoBERTa*, ktorý dosiahol precision 0.7899, recall 0.7415 a F1-score 0.7649. Tretie miesto patrí modelu RoBERTa s hodnotou F1-score 0.7569.



Obr. 13.1: Experiment:

Celkovo možno zhrnúť, že pre tento dataset dosiahol SlovakBERT najlepšie výsledky a je preto najlepším modelom pre klasifikáciu textu týkajúceho sa COVID-19 v slovenskom jazyku. Ak sa pozrieme na výsledky jednotlivých modelov bližšie môžeme si povšimnúť nasledovné zistenia. V nasledujúcej časti sa pozrieme na výsledky týchto modelov bližšie a zhodnotíme úspešnosť aj s odôvodnením tohto výsledku.

Na prvom mieste sa umiestnil SlovakBERT, veľký lingvistický model určený pre slovenský jazyk. Aj napriek menšiemu počtu parametrov dokázal SlovakBERT zvíťaziť nad modelmi RoBERTa a jeho multilingválnou verziou, vďaka tomu, že je prvým modelom s architektúrou Transformer určeným pre slovenský jazyk a využil plne pochopenie slovenských viet oproti konkurencii.

Týmto umiestnením sa potvrdilo, že aj v dnešnej dobe sú špeciálne modely pre jednotlivé jazyky potrebné a umožňujú kvalitnejšie predikcie v danom jazyku. Tento fakt je umocnený ešte kvalitou dát s ktorými pri riešení daného lingvistického problému pracujeme. V tomto prípade maličký slovenský dataset v ktorom zahraničné modely nemôžu prekonať fine-tuning slovenského modelu pre slovenský jazyk, ktorý je už v základe pripravený na prácu s textami v tomto jazyku a tým pádom si vie lepšie poradiť s jednotlivými úskaliami slovenského jazyka pri rišení rôžnych lingvistických úloh.

Za týmto modelom sa umiestnili všetky tri verzie RoBERTa modelu, ktorých rozdiely boli minimálne. Toto umiestnenie oproti modelu BERT/mBERT nás neprekvapuje. Je spôsobené väčším počtom parametrov pre tieto modely oproti základnej verzii BERT, a väčšou množinou dát na ktorej boli predtrénované. Tieto tri modely dosahujú skoro totožné výsledky. Ukazuje sa, že pre tento model nehrá rolu pre multilingválnu verziu, v ktorom z podporovaných jazykov je dataset. Taktiež sa ukazuje, že rozsiahlejší model kompenzuje stratu v preklade. Na druhú stranu, táto strata je markantná oproti natívnej verzii SlovakBERT.

Modely BERT sa umiestnili na konci rebríčka. Konkrétne anglická verzia BERT, hneď za RoBERTa modelmi. Táto strata je spôsobená ako sme už spomenuli menším počtom parametrov modelu ako aj nutnosťou prekladu textov. Tento proces môže viesť k strate niektorých informácií, ktoré sú obsiahnuté v pôvodnom texte, alebo k ich nahradeniu sémanticky podobnými verziami. V kontexte detekcie falošných správ môže dôjsť k stratám malých odchýlok, ktoré odlišujú pravdivé správy od tých falošných.

Vyššia úspešnosť modelu BERT je spôsobená tým, že multilingválna verzia mBERT resp. mBERT* je natrénovaná na dátach, z ktorých napríklad tvorili články v slovenskom jazyku približne jedno percento. Na druhú stranu, pôvodný BERT je natrénovaný čisto na anglických textoch a nástroje na preklad textov sú v dnešnej dobe zväčša natrénovane na duálnych korpusoch, a nie na multilingválnych dátach, nevedia teda prekladať medzi ľubovoľnou dvojicou jazykov, ale na druhú stranu obmedzenie prekladu na dva jazyky zvyšuje jeho presnosť.

Na konci nášho rebríčka sa teda umiestnil multilingválny mBERT. Tento model dopláca na svoju robustnosť, kedy vie pracovať s textovými dátami vo viac ako 100 jazykoch. Dvojica mini-experimentov, pozostávajúca zo súťaže medzi dvoma verziami multiligualného BERT-u potvrdila, že prekladom sa strácajú niektoré vlastnosti pôvodného textu. Verzia mBERT*, ktorá bola použitá s preloženým slovenským datasetom (rovnako ako BERT stráca oproti SlovakBERT) má nižšiu úspešnosť ako mBERT, ktorý bol použitý priamo na slovenskom texte. Tento výsledok iba potvrdil, že použitie textových dát v pôvodnom jazyku zvyšuje úspešnosť modelu. V pôvodnom jazyku textu sú všetky odchýlky, ktoré odlišujú fake-news od pravdivých správ, kdežto pri preklade sa niektoré z týchto informácií z textu vytratia a jemné rozdiely medzi niektorými fake-news a pravdivými správami sa zotrú.

Výsledky tohto experimentu ukazujú, že v prípade, že pre daný jazyk neexistuje verzia modelu BERT, je možné použiť multilingválne alebo zahraničné verzie veľkých lingvistických. V prípade, že je daný jazyk podporovaný niektorým z multilingválnych modelov, je možné ich použiť priamo. Avšak, v prípade, že je nutný preklad textov, môže byť tento proces náročný, a aj úspešnosť môže byť nižšia v porovnaní s natívnymi modelmi.

13.2 Fine-tuning na anglických datasetoch

Táto skupina experimentov je zameraná na fine-tuning modelov na jednom z anglických datasetov a iba následná evaluácia prebehne na slovenskom testovacom datasete t.j. na identických článkoch ako v predošlom experimente. Pre modely BERT, RoBERTa, mBERT* a XLM-RoBERTa* na preloženej verzii pomocou automatického prekladu v Python-e knižnicou *googletrans*. Evaluácia pre mBERT respektíve pre XLM-RoBERTa prebehne na pôvodnom nepreloženom datasete. Na druhú stranu pre slovenskú verziu SlovakBERT modelu sú anglické datasety LIAR a COVID19 FN preložené do slovenského jazyka.

13.2.1 LIAR

V tomto dielčom experimente prebehol fine-tuning modelov na anglickom datasete LIAR z kapitoly 9.1.1. Cieľom tohto experimentu je overiť hypotézu vplyvu témy fake-news na výslednú detekciu.

Ak sa pozrieme na výsledky v tabuľke 13.2 a porovnáme ich s výsledkami z prvého experimentu uvedenými v tabuľke 13.1. Zistíme, že fine-tuning iba na anglickom datasete LIAR bez fine-tuningu na slovenských dátach nepostačuje. A takto natrénované modely sú slabšie ako najhorší model v predošlom experimente 13.1. Takmer všetky modely predikovali každú správu ako falošnú.

Experiment	epochs	precision	recall	accuracy	f1-score
BERT	20	0.5017	1.0000	0.5017	0.6682
mBERT*	20	0.5017	1.0000	0.5017	0.6682
mBERT	20	0.5017	1.0000	0.5017	0.6682
RoBERTa	20	0.5368	0.8435	0.5563	0.6561
XLM-RoBERTa*	20	0.5017	1.0000	0.5017	0.6682
XLM-RoBERTa	20	0.5017	1.0000	0.5017	0.6682
SlovakBERT	20	0.5017	1.0000	0.5017	0.6682

Tabuľka 13.2: Experiment: Fine-tuning na anglickom dadasete LIAR.

V porovnaní s predošlým experimentom 13.1, kde sme porovnávali model RoBERTa s rovnakou architektúrou, sme zistili, že v aktuálnom experimente dosiahol tento model nižšiu úspešnosť o 9%. Toto zhoršenie výkonu nie je spôsobené jazykom, v ktorom sú textové dáta, ako sme ukázali v predošlom experimente, kde sme preukázali, že pri rovnakej téme dát je možné chybu prekladu takmer zanedbať. Preto môžeme text považovať za napísaný v preloženom jazyku. Vzhľadom na výsledky v tabuľke 13.1, kde sme zistili, že pre RoBERTa nezáleží na jazyku dát, musí zvýšenie chyby v aktuálnom experimente súvisieť s odlišnou témou článkov v trénovacom a testovacom datasete.

Multilingválne modely sú na tom ešte horšie. Tie sa vo fáze fine-tuningu adaptovali nie len na tému dát ale aj na anglický jazyk. Neboli na slovenskom testovacom datasete schopné správnej klasifikácie.

Nezhoda témy sa potvrdila najvýraznejšie na slovenskom modeli SlovakBERT, ktorého úspešnosť klesla o viac ako 20%. Horšie na tom je ale to, že model predikuje prakticky celú množinu ako falošnú a teda zhoršenie je rovné klasifikátoru, ktorý predikuje iba falošné správy.

Na tomto experimente sa ukázalo, že fake-news s rôznou témou majú iný textový charakter. Model sa nedá natrénovať iba na jednej doméne fake-news a zovšeobecniť ho na detekciu v akejkoľvek téme.

13.2.2 COVID19 FN

V tomto teste prebehol fine-tuning modelov na anglickom datasete COVID19 FN z kapitoly 9.1.2. Keďže sme v predošlom experimente ukázali, že téma článkov ma obrovský vplyv na štruktúru textov a teda aj na následnú klasifikáciu. V nadväznosti na to, sme sa rozhodli týmto experimentom overiť, či falošne správy ohľadom rovnakej témy z rôznych kútov sveta možno spájať do spoločných datasetov, pretože majú spoločný charakter nosnej správy.

Ak porovnáme výsledky dosiahnuté modelmi na ktorých prebehol fine-tuning pomocou COVID19 FN v tomto experimente s výsledkami tých, ktoré boli prispôsobené v predošlom experimente za pomoci LIAR datasetu, tak sú tieto výsledky pre väčšinu modelov podobné. Tu sa ukazuje, že každá krajina má určite špecifiká pri písaní fake-news. Rovnaká téma článkov pomohla týmto modelom zvýšiť úspešnosť, ale iba miernym spôsobom. To iba potvrdilo fakt, že v rámci rovnakej témy sa určité fake-news šíria v svojich jazykových mutáciách naprieč rôznymi jazykmi. Takéto fake-news musia byť teda všeobecne platné kdekoľvek na svete a

nebudú sa vzťahovať na lokálnu situáciu. Napríklad, pre pandémiu Covid-19 to sú falošné správy ohľadom vakcinácie, úmrtnosti, liečby atď.

Experiment	epochs	precision	recall	accuracy	f1-score
BERT	20	0.5296	0.9728	0.5529	0.6859
mBERT*	20	0.5603	0.8844	0.5939	0.6860
mBERT	20	0.5017	1.000	0.5017	0.6682
RoBERTa	20	0.5477	0.8979	0.5768	0.6804
XLM-RoBERTa*	20	0.5524	0.9320	0.5870	0.6937
XLM-RoBERTa	20	0.5017	1.0000	0.5017	0.6682
SlovakBERT	20	0.5991	0.8844	0.6451	0.7143

Tabuľka 13.3: Experiment: Fine-tuning na anglickom datasete COVID19 FN.

Na druhú stranu, týmto experimentom sme ukázali, že fine-tuning modelu na detekciu fake-news iba na datasete, ktorý neobsahuje špecifiká danej lokálnej komunity je nepostačujúci. Modely, ktoré sú prispôsobené priamo na dátach z tejto komunity (oblasti) majú výsledky minimálne o 10% kvalitnejšie, a k tomu odpadá aj nutnosť prekladu.

13.3 Spojenie datasetov

V poslednom experimente plynulo nadviažeme na tie predošlé. Skočili sme fine-tuningom modelov SlovakBERT, BERT, mBERT, mBERT*, respektíve RoBERTa, XLM-RoBERTa a XLM-RoBERTa* na anglických datasetoch LIAR a COVID19 FN. Takto prispôsobené modely pre úlohu fake-news sme použili na klasifikáciu fake-news ale zo Slovenska, aby sme ukázali že kontext danej správy sa prekladom nemení.

Tento experiment však môže pokračovať aj ďalším krokom, a tým je opätovný fine-tuning, ale tentokrát na slovenskom trénovacom datasete.

Až po fine-tuningu na tomto datasete uskutočníme evaluáciu na slovenskej testovacej množine. Výsledky tohto procesu sú uvedené v tabuľkách 13.4 a 13.5.

13.3.1 LIAR

Fine-tuning u modelov v tejto časti najskôr prebehol na LIAR datasete, následne opätovný fine-tuning ale už na slovenskom trénovacom datasete. Výsledky po evaluácii na testovacom datasete sú uvedené v tabuľke 13.4.

Pre modely mBERT* a XLM-RoBERTa* prebehne fine-tuning najskôr na anglickom datasete LIAR a následne na preloženom trénovacom slovenskom datasete Downsampled SlovakCovid19 FN. Taktiež evaluácia prebehla na preloženej testovacej slovenskej sade. Na druhej strane pre SlovakBERT sme LIAR dataset preložili do slovenského jazyka, podobne ako v predošlom teste.

Vyhodnotením výsledkov, ktoré sú v tabuľke 13.4 sme dospeli k nasledujúcim záverom.

Oproti fine-tuningu iba na LIAR datasete a následnom vyhodnotení už na slovenskom testovacom datasete došlo k zlepšeniu u anglických modelov respektíve

Experiment	epochs	precision	recall	accuracy	f1-score
BERT	20	0.7698	0.7279	0.7543	0.7483
mBERT*	20	0.8333	0.0340	0.5119	0.0653
mBERT	20	-	0.0000	0.4983	-
RoBERTa	20	0.8000	0.6803	0.7543	0.7353
XLM-RoBERTa*	20	0.5017	1.0000	0.5017	0.6682
XLM-RoBERTa	20	0.5017	1.0000	0.5017	0.6682
SlovakBERT	20	-	0.0000	0.4983	-

Tabuľka 13.4: Experiment: Fine-tuning (LIAR)

multilingválnych modelov na preložených dátach. Toto zlepšenie u týchto modelov bolo očakávané, keďže sme modely ešte dotrénovali na slovenských trénovacích dátach. Na druhú stranu neprekonali modely, ktoré boli trénované iba na slovenskom datasete. Táto skutočnosť je znovu spôsobená rôznou témou článkov, kedy LIAR správy skresľujú výsledok a prinášajú šum do modelu. Tento šum má za následok vyššiu falošnú pozitivitu.

V tomto experimente sme použili multilingválne modely, ktoré sme nezaťažili prekladom slovenského datasetu do angličtiny. Tieto modely sme najskôr fine-tunovali na anglickom LIAR datasete a následne sme ich trénovali na slovenskom datasete. Avšak, v porovnaní s predchádzajúcim experimentom sme zaznamenali ešte väčšie zhoršenie úspešnosti modelov na nepreloženom slovenskom datasete. Zhoršenie je spôsobené spôsobom fine-tuningu, kde si v prvom kroku najskôr modely prispôsobili svoje váhy a rozhodnutia anglickému jazyku. V druhom kroku už boli ale donútené pracovať so slovenskou sadou na ktorej prebehlo aj vyhodnotenie. Týmto striedaním jazykov sa zanieslo do fine-tuningu ešte väčšie množstvo šumu. Ako sa ukázalo týmto testom, je vhodnejšie trénovacie dáta automaticky preložiť do jedného z jazykov a následne na nich spustiť fine-tuning. Takto natrénované modely poskytujú lepšie výsledky.

V tomto experimente sa ukázalo, že SlovakBERT nedokázal správne klasifikovať testovacie správy a označil ich všetky ako pravdivé. Toto zlyhanie bolo zapríčinené rozdielnou témou trénovacích dát a aj chybou prekladu anglického datasetu do slovenského jazyka.

Z tohto experimentu vyplýva, že aj keď je známe, že väčší objem dát môže vylepšiť výkon modelu, v prípade fake-news je kľúčové zohľadniť tému testovaných falošných správ a tiež dôležitosť jazykových špecifík cieľového jazyka, v ktorom sú texty napísané.

13.3.2 COVID19 FN

Fine-tuning u modelov v tejto časti, podobne ako pre LIAR, najskôr prebehol na COVID19 FN datasete a následne opätovný fine-tuning, ale už na slovenskom trénovacom datasete. Výsledky po evaluácii na testovacom datasete sú uvedené v tabuľke 13.5. Podobne ako v predošlom experimente, aj v tomto pre multilingválne modely prebehol test ako s preloženými datasetmi do anglického jazyka, tak aj na kombinácii angličtiny a slovenčiny podľa toho v akom jazyku bol daný dataset pôvodne.

Experiment	epochs	precision	recall	accuracy	f1-score
BERT	20	0.7450	0.7551	0.7474	0.7500
mBERT*	20	0.7928	0.5986	0.7201	0.6822
mBERT	20	0.8062	0.7075	0.7679	0.7536
RoBERTa	20	0.7410	0.8367	0.7713	0.7859
XLM-RoBERTa*	20	0.7260	0.7211	0.7235	0.7235
XLM-RoBERTa	20	0.7313	0.6667	0.7099	0.6975
SlovakBERT	20	0.8095	0.8095	0.8089	0.8095

Tabuľka 13.5: Experiment: Fine-tuning (COVID19 FN)

Pre slovenský model SlovakBERT sme opätovne preložili COVID19 FN do slovenského jazyka ako v predošlom prípade LIAR datasetu.

Tento test zavŕšuje sadu experimentov pomocou modelov založených na architektúre Tranformer na DownSampled SlovakCoid19 FN dataste. Tento test sme založili na rovnakom princípe ako predošlý na LIAR datasete len s tým cieľom, že ich chceme porovnať navzájom a zistiť, či rovnaká téma medzi datasetmi v iných jazykoch môžeme pomôcť modelom pri detekcii falošných správ. Ako vidíme v tabuľke 13.5, rovnaká téma cudzojazyčným datestov pomohla oproti LIAR experimentu. Došlo k veľkému zlepšeniu medzi zahraničným datasetom s rovnakou a odlišnou témou. Všetky modely okrem toho dosiahli zlepšenie oproti tréningu iba na zahraničných dátach.

To znamená, že aj malý dataset v cieľovom jazyku s rovnakou témou falošných správ dokáže pomôcť zlepšiť predikciu týchto modelov, ktoré sú trénované na zahraničných dátach. Na druhú stranu, tento zahraničný dataset musí byť v rovnakej tematickej oblasti ako cieľový. Toto zistenie vyplynulo z porovnania testov LIAR a COVID19 FN datasetov.

Treba tiež podotknúť, že čisto slovenský model, ktorý bol natrénovaný iba na slovenskom datasete dosiahol napriek tomu najlepšie výsledky. Tento stav je spôsobený špecificitou fake-news pre Slovensko. Táto špecificita je daná malým množstvom médií, ktoré šíria na Slovensku falošné správy a tým sa odlišujeme od zahraničia, kde je týchto zdrojov väčšie množstvo a je možné zovšeobecniť detekciu falošných správ.

Tieto zistenia je možné interpretovať nasledovne, ak pre daný jazykový korpus existuje predtrénovaný model, je optimálne ho použiť na detekciu falošných správ. Ak tento model pre daný jazykový korpus neexistuje, potom je lepšie použiť metódu automatického prekladu v spojení s modelom, ktorý je určený pre daný preložený jazyk. Najmenej vhodnou alternatívou je použite multilingválych modelov napríklad typu XLM-RoBERTa v danom jazyku ak ho podporuje, alebo automaticky preložiť textu do podporovaného jazyka. Na druhú stranu, ukazuje sa, že nie je vhodné na fine-tuning použiť inú tematickú oblasť akej sa týka cieľová oblasť fake-news. Podobne aj kombinácia viacerých rôznych jazykových korpusov zanáša šum do modelov.

13.4 SlovakCovid19 FN

Poslednou sadou experimentov v tejto diplomovej práci, ktoré sme vykonali pomocou modelov založených na architektúre Transformer, bola ich implementácia pre celý slovenský dataset na základe výsledkov predošlých testov. Ako sa ukázalo v predošlých experimentoch, najvyššiu úspešnosť mala slovenská verzia modelu BERT, ktorá bola natrénovaná čisto na slovenskom datasete. Preto sme sa v tomto poslednom teste rozhodli otestovať práve túto kombináciu modelu a datasetu a vyhodnotiť ich úspešnosť.

Na tejto kombinácii modelu a datasetu sme vykonali radu experimentov s rôznymi modifikáciami ako modelu tak aj trénovania, ktoré sú súčasťou tejto práce v priložených súbor. Modifikácia modelu SlovakBERT prebiehala pripojením ďalších vrstiev za tento model. Týmito vrstvami boli CNN, LSTM, RCL, plnepreposjené vrstvy, dropout vrstva a rôzne iné možnosti. Najúspešnejšiu verziu tohto modelu si uvedieme v tejto časti kapitoly aj s výsledkami, ktoré dosiahol na tomto datasete. Podobne ako v predošlých experimentoch bol celý dataset rozdelený na tri časti v pomere 6:2:2 (train, valid, test).

Pri experimentoch sme spravili radu testov, ktoré overovali rôzne kombinácie sietí a preprocesingu. Výslednú architektúru, ktorá dosiahla najlepšie výsledky pri detekcii fake-news na celom slovenskom datasete, sme pre model SlovakBERT naimplementovali v jazyku Python pomocou knižnice *tensorflow*, a je nasledovná:

```
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_ids (InputLayer)      [(None, 512)]             0

 attention_mask (InputLayer)  [(None, 512)]            0

 roberta (TFRobertaMainLayer)  TFBaseModelOutputWi  124644864
                             thPoolingAndCrossAt
                             tentions(last_hidde
                             n_state=(None, 512,
                              768),
                              pooler_output=(Non
                             e, 768),
                              past_key_values=No
                             ne, hidden_states=N
                             one, attentions=Non
                             e, cross_attentions
                             =None)

 spatial_dropout1d (SpatialDrop  (None, 512, 768)    0
 out1D)

 global_average_pooling1d (Glob  (None, 768)         0
 alAveragePooling1D)

 dense (Dense)               (None, 128)               98432
 dense_1 (Dense)             (None, 64)                8256
 dense_2 (Dense)             (None, 32)                2080
 dense_3 (Dense)             (None, 16)                528
 dropout_37 (Dropout)        (None, 16)                0
 dense_4 (Dense)             (None, 2)                 34
=================================================================
```

```
Total params: 124,754,194
Trainable params: 124,754,194
Non-trainable params: 0
----------------------------------------------------------------
```

Listing 13.8: SlovakBERT - SlovakCovid19 FN

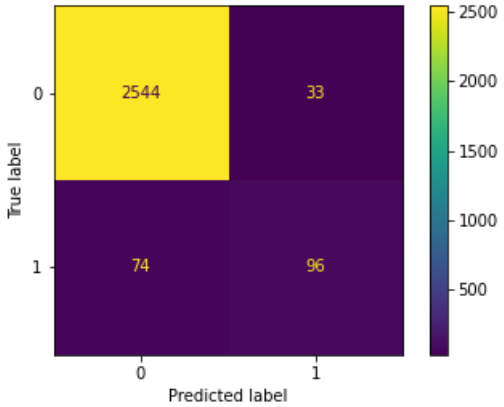
SlovakBert rozšírený o tieto klasifikačné vrstvy dosiahol výsledky, ktoré sú uvedené v tabuľke 13.6 a na ilustrácii konfúznej matice 13.2.

Experiment	epoch	precision	recall	accuracy	f1-score
SlovakBERT	20	0.7442	0.5647	0.9610	0.6421

Tabuľka 13.6: Experiment: SlovakBERT (SlovakCovid19 FN).

Finálna verzia SlovakBERT modelu v tejto implementácii dosiahla najlepšie výsledky na celom slovenskom datasete z pomedzi všetkých modelov založených na architektúre Transformer. Jej úspešnosť je najlepšia spomedzi všetkých spôsobov automatickej detekcie falošných správ na slovenských textoch, ktoré sme v tejto práci testovali na tomto datasete.

Prekonáva základné metódy strojového učenia aj hlboké neurónové siete vo všetkých nami sledovaných metrikách. Podobne dosahuje najlepšie výsledky z rady experimentov založených na architektúre Transformer, ktoré sme taktiež vykonali na tomto datasete.



Obr. 13.2: Experiment: SlovakBERT (SlovakCovid19 FN).

Z konfúznej matice tohto modelu môžeme vidieť veľkú nevyváženosť v celom datasete a možný potenciál zlepšenia výseledkov tohto modelu na základe predošlých experimentov pri zlepšení celého slovenského datasetu.

13.5 Zhrnutie

Posledná sada experimentov sa zamerala na veľké lingvistické modely založené na architektúre Transformer z kapitoly 8. Výkonnosť jednotlivých modelov rovnako ako v predošlých testoch bola porovnaná na SlovakVodi19 FN datasete, podobne ako na jeho DownSampled verzii. Pre fine-tuning boli použité aj zahraničné dátové sady, konkrétne LIAR a Covid19 FN v anglickom jazyku.

Touto sadou testov sme chceli overiť ako úspešnosť modelov založených na architektúre BERT, tak aj vplyv rôznorodosti datasetov, ktoré sa líšili v téme falošných správ, jazyku, prípadne vplyv veľkosti datasetu.

Pre overenie týchto všetkých možných kombinácii sme v tejto sérii testov vykonali bezmála 50 experimentov s rôznymi modelmi v kombinácii s voľbou parametrov a datasetom. Výsledkom týchto experimentov sú zistenia, ktorých zhrnutie uvedieme v nasledujúcich odstavcoch.

Prvým zistením a zároveň najzásadnejším, ktoré vyplynulo z rady našich experimentov je sila natívnych modelov. Ukázalo sa, že aj keď multilingválne modely prípadne cudzojazyčné spojené s automatickým prekladom poskytujú dobrú alternatívu v prípade nedostupnosti natívneho modelu. Na druhú stranu, tieto modely v prípade existencie modelu založeného na architektúre BERT (prípadne inej založenej na architektúre Transformer) priamo pre cieľový jazyk, nedokážu tento model prekonať a nedosahujú ani zďaleka tak dobré výsledky.

Druhým zistením, ale nie menej dôležitým, je vplyv témy falošných správ na štruktúru textových dát. Ukázalo sa, že tematicky rôzne falošné správy nezdieľajú také množstvo lingvistických znakov. To znamená, že pre predikciu fake-news na základe textových dát je nutné pri tréningu zachovať tematiku falošných správ na základe testovacej množiny. Obzvlášť pri zahraničných zdrojoch v spojení s prekladom. Inak je úspešnosť takého modelu zretelne znížená.

Posledným výsledkom týchto experimentov je vplyv kvality datasetu na celkovú predikciu klasifikátora. Vplyv datovej sady na celkovú úspešnosť modelu je markantná, a ukazaje sa, že kvalita datasetov pre detekciu falošných správ pre slovenský jazyk je nedostatočná.

Zhrnutím našich poznatkov je nutnosť skvalitnenia trénovacích dát pre detekciu fake-news na Slovensku prípadne pre iné jazykové korpusy a zároveň použitie, respektíve vytvorenie predtrénovaných veľkých jazykových modelov pre konkrétny jazyk. Tieto dve zistenia môžu dopomôcť k zlepšeniu automatickej detekcie falošných správ na Slovensku, aj vo svete.

Záver

V rámci tejto práce sme sa zaoberali problematikou automatickej detekcie falošných správ (fake-news) v slovenčine z pohľadu klasifikácie textu. Fake-news, teda úmyselne nepravdivé alebo skreslené informácie, sa v poslednej dobe na internete vyskytujú čoraz častejšie a preto je dôležité naučiť sa ich automaticky detegovať. Práca je jedinečná v tom, že spravodajské články boli spracované v slovenskom jazyku, pre ktorý ešte takto rozsiahla sada experimentov nebola vykonaná. Väčšina doterajších štúdii, ktoré sa zaoberali touto problematiku, bola buď určená pre cudzojazyčné dátové sady, typicky pre angličtinu, alebo štúdie určené pre slovenčinu iba popisovali vytvorenie datasetu so základnou sadou experimentov.

Naším hlavným cieľom bolo vytvorenie nástroja, ktorý by dokázal detegovať falošné správy na slovenských textoch, a to nielen z novinových článkoch, ale aj v príspevkoch na sociálnych sieťach. Okrem toho sme si stanovili viacero dielčich krokov, ktoré nám mali pomôcť tento cieľ dosiahnuť. Jedným z nich bolo vytvorenie vlastného datasetu pre detekciu falošných správ v slovenskom jazyku. Následným krokom bolo vytvorenie a otestovanie základných modelov pre rôzne metódy strojového učenia s cieľom poskytnúť ucelený pohľad na automatickú detekciu falošných správ v slovenskom jazyku. Tieto kroky boli dôležité a ich splnenie nám umožnilo lepšie porozumieť procesu automatickej detekcie fake-news a taktiež položilo základ pre ďalšie výskumy v tejto oblasti.

Na začiatku práce ešte pred samotnou tvorbou klasifikátora bolo nevyhnutné nájsť dostatočne veľký dataset pre trénovanie a testovanie detekčných modelov v slovenskom jazyku. Avšak, získanie tohto datasetu bolo výzvou, pretože v tom čase nebola verejne dostupná žiadna slovenská dátová sada. Nakoniec sa nám podarilo získať základnú dátovú sadu od tímu Sarnovského na Technickej univerzite v Košiciach. Umožnil nám týmto krokom prístup k prvému slovenskému datasetu na detekciu falošných správ v tematike Covid-19. Na základe tejto tematiky slovenskej dátovej sady sme vybrali vo vzťahu k nej aj ostatné cudzojazyčné datasety na detekciu falošných správ. Konkrétne sme vybrali anglickú dátovu sadu COVID19 FN, ktorá má rovnakú tematiku článkov ako slovenský dataset. Druhým anglickým datasetom, ktorý sme v tejto práci použili je LIAR, ktorý v sebe obsahuje články z politickej oblasti. Táto dátová sada bola vybraná na otestovanie spoločných vlastností fake-news s rôznou témou článkov.

Vykonali sme viac ako osemdesiat experimentov na zistenie vhodnej kombinácie predspracovania textu a modelu strojového učenia. Taktiež sme vykonali radu testov s cudzojazyčnými (anglickými) datasetmi v kombinácii so slovenskou dátovou sadou, na ktorých sme overili prenositeľnosť znakov falošných správ medzi jazykmi respektíve témami. Po niekoľkých experimentoch s rôznymi modelmi strojového učenia sme zistili, že náš pôvodný dataset bol nevyvážený a len ťažko riešiteľný pre základné modely. Preto sme sa rozhodli vytvoriť vlastnú zmenšenú verziu datasetu s vyváženými triedami, ktorá by mohla slúžiť ako referenčný benchmark pre naše ďalšie experimenty a ako trénovacia sada pre rôzne modely. Tento nový dataset bol vytvorený z pôvodného datasetu s manuálne overenými anotáciami, ktorý sme zmenšili a vyvážili a v poslednom kroku sme ho použili ako referenčný bod pre všetky naše experimenty.

Následne sme na oboch slovenských datasetoch otestovali základnú sadu experimentov. V nej sme vykonali testy za pomoci jednoduchých modelov strojového učenia ako Bayesov model, rozhodovacích strom, random forest, k-nn a XGB. Na základe výsledkov krížovej validácie jednotlivých metód sme vytvorili optimalizovaný baseline model za pomoci XGB, ktorý dosiahol najlepšie výsledky $F1_{downsampled} = 0.7065$ pre zmenšenú verziu datasetu a $F1 = 0.5831$ pre celý pôvodný dataset. V ďalšom kroku sme otestovali komplexnejšie modely založené na architektúre neurónových sietí. Konkrétne CNN, RNN a RCNN, ktorých výsledky sme porovnali s baseline modelom, ale aj medzi sebou. Všetky neurónové siete v porovnaní so základnou sadou experimentov dosiahli lepšie výsledky. Najkvalitnejšie výsledky pre oba slovenské datasety dosiahla neurónová sieť LSTM. Na druhú stranu už v tomto experimente sa ukázalo, že celý slovenský dataset by potreboval kvalitnejšie dáta z dôvodu nízkeho obsahu falošných správ.

Najrozsiahlejšia časť práce sa zaoberá experimentami s modelmi založenými na architektúre Transformer. Modely založené na tejto architektúre, ktoré sme otestovali boli BERT, mBERT, Roberta, XLM-Roberta a SlovakBERT. Tieto modely sme otestoval na oboch slovenských datasetoch, ako aj na kombinácii s anglickými verziami datasetov. Závery zo sady experimentov poukázali na to, že modely s komplexnejšou architektúrou sa dokážu lepšie vysporiadať so stále nedostatočnou slovenskou dátovou sadou. Zároveň sa však ukázala potreba vytvárania kvalitnejších datasetov.

Na druhú stranu sme zistili, že kombinácia slovenských datasetov s datasetmi v iných jazykoch s cieľom rozšírenia obsahu textov má svoje obmedzenia. Pri písaní falošných správ sa prejavuje špecifickosť slovenského jazyka, ktorá sa stráca pri preklade do iných jazykov. Rovnako spracovanie multilingválnymi modelmi nevie dostatočne podchytiť túto jemnú odchýlku. Tento efekt síce nie je až taký výrazný v porovnaní s multilingválnymi modelmi, no s prvým slovenským modelom založeným na architektúre Transformer je stále významný. Toto zistenie potvrdilo nutnosť aj naďalej vytvárať špecifické modely pre každý konkrétny jazykový korpus.

Pri testovaní kombinácie slovenských a zahraničných dátových sád sme ďalej zistili, že vplyv témy článkov na trénovanie a testovanie modelov je významný. Zistili sme, že trénovanie modelov na jednej oblasti falošných správ a ich následné použitie v inej oblasti nie je možné. Možno predpokladať, že podobné témy alebo dokonca rovnaký jazyk môžu byť zahrnuté do jedného datasetu. Avšak, datasety s falošnými správami pre rôzne jazyky a s rôznymi témami nie sú zlúčiteľné a prispievajú k veľkému množstvu odlišností a šumu počas učenia modelov.

V rámci poslednej sady experimentov boli testované rôzne modely založené na architektúre Transformer a ich výsledky boli zhodnotené. Zistilo sa, že modely BERT a mBERT dosahovali všeobecne nižšiu úspešnosť ako ostatné testované modely. Tento fakt sa môže vysvetliť nižším počtom parametrov a špecifikami slovenského jazyka. Modely RoBERTa a XLM-RoBERTa dosiahli lepšie výsledky ako BERT a mBERT, ale na druhú stranu najúspešnejším modelom sa stal SlovakBERT na trénovaný iba na slovenskej dátovej sade, ktorý má rovnaký počet parametrov ako BERT. Tento model v kombinácii s datasetom dosiahol najlepšie výsledky na testovacej množine pre obe slovenské datasety, pri zmenšenom datasete dosiahol hodnoty $F1_{downsampled} = 0.8824$ a $acc_{downsampled} = 0.8771$ (o 12% viac), zatiaľ čo na celej sade dosiahol prekvapivo dobré výsledky $F1 = 0.6421$

a $acc = 0.9610$ (acc o 2% viac ako baseline model a F1 zhruba o 20%).

SlovakBert model trénovaný na čisto slovenskom datasete dosiahol pozoruhodné výsledky v detekcii fake-news. Jednou z jeho výhod v porovnaní s cudzojazyčnými modelmi je, že nie je potrebné prekladať slovenské texty, čo eliminuje šum spôsobený prekladom a zlepšuje presnosť klasifikácie. Avšak, je nutné počítať s tým, že ak takýto model neexistuje pre jazyk, jeho pretrénovanie je výpočetne náročne. Na druhú stranu, ak takýto model existuje potom nie je nutné prekladať texty a zvyšuje to ako presnosť modelu, tak aj výslednú rýchlosť klasifikácie. Je možné argumentovať, že použitie multilingválnych modelov eliminuje potrebu prekladu textov, čím sa ušetrí čas a úsilie pri detekcii falošných správ. Tieto modely sú však zvyčajne trénované na korpusoch obsahujúcich viacero jazykov a množina dát pre konkrétny jazyk, na ktorý sa chceme zamerať, môže byť minimálna. Toto má častokrát za následok zníženú presnosť týchto modelov pri detekcii falošných správ.

V tejto práci sme položili základný kameň pre detekciu falošných správ v slovenskom jazyku, na ktorej výsledkoch sa v budúcnosti dá stavať. Dúfame, že výsledky práce poslúžia ako inšpirácia pre ďalší výskum v oblasti klasifikácie textu v oblasti fake-news.

Budúca práca

Detekcia falošných správ je v dnešnej dobe čoraz väčší a aktuálnejší problém, ktorý je nutné stále riešiť. V nadväznosti na túto prácu by bolo možné pokračovať rôznymi smermi. Jedným z možných smerovaní pre zlepšenie tejto klasifikácie falošných správ je vytvorenie kvalitnejších a rozsiahlejších dátových sád z rôznych tematických oblastí v slovenskom jazyku.

Ďalšou možnosťou je použitie na transfer a fine-tuning iné jazyky ako angličtina, napríklad čeština, poľština alebo iné slovanské jazyky, ktoré sú tomuto jazyku bližšie, čo by mohlo viesť na zlepšenie a ukázanie zastúpení jazykov so spoločnými znakmi.

Iným smerom môže byť kombinácia nelingvistických a lingvistických metód pre detekciu falošných správ. Napríklad kombinácia textu a multimediálneho obsahu článku pre zlepšenie detekcie. Prípadne použitie metadát s čím súvisí vytvorenie datasetu, ktorý v sebe zahŕňa aj tieto informácie.

Poslednou, ale nie menej dôležitou možnosťou, je použitie už zistených poznatkov z tejto práce v praxi. Počas písania diplomovej práce som nadviazal kontakt so spoločnosťou KInIT, ktorá vyvinula prvý slovenský veľký jazykový model SlovakBERT a zaoberá sa problematikou falošných správ. Táto spoločnosť prejavila záujem o výsledky tejto diplomovej práce, keďže sa touto problematikou aktívne zaoberá.

Zoznam použitej literatúry

[1] The legal framework to address “fake news”: possible policy actions at the eu level, volume = , author = Andrea Renda (CEPS - Centre for European Policy Studies and College of Europe, editor = , year = 2018.

[2] Martin Sarnovskỳ, Viera Maslej-Krešňáková, and Klaudia Ivancová. Fake news detection related to the covid-19 in slovak language using deep learning methods. *Acta Polytechnica Hungarica*, 19(2):43–57, 2022.

[3] Parth Patwa, Shivam Sharma, Srinivas Pykl, Vineeth Guptha, Gitanjali Kumari, Md Shad Akhtar, Asif Ekbal, Amitava Das, and Tanmoy Chakraborty. Fighting an infodemic: COVID-19 fake news dataset. In *Combating Online Hostile Posts in Regional Languages during Emergency Situation*, pages 21–29. Springer International Publishing, 2021.

[4] Gautam Kishore Shahi and Durgesh Nandini. Fakecovid - A multilingual cross-domain fact check news dataset for COVID-19. *CoRR*, abs/2006.11343, 2020.

[5] Limeng Cui and Dongwon Lee. Coaid: COVID-19 healthcare misinformation dataset. *CoRR*, abs/2006.00885, 2020.

[6] William Yang Wang. "liar, liar pants on fire": A new benchmark dataset for fake news detection. *arXiv preprint arXiv:1705.00648*, 2017.

[7] Anna Glazkova, Maksim Glazkov, and Timofey Trifonov. g2tmn at constraint@AAAI2021: Exploiting CT-BERT and ensembling learning for COVID-19 fake news detection. In *Combating Online Hostile Posts in Regional Languages during Emergency Situation*, pages 116–127. Springer International Publishing, 2021.

[8] Martin Müller, Marcel Salathé, and Per E Kummervold. Covid-twitter-bert: A natural language processing model to analyse covid-19 content on twitter, 2020.

[9] Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Ruslan Salakhutdinov, and Quoc V. Le. Xlnet: Generalized autoregressive pretraining for language understanding, 2019.

[10] Zekeriya Anil Guven, Banu Diri, and Tolgahan Cakaloglu. Comparison of topic modeling methods for type detection of turkish news. In *2019 4th International Conference on Computer Science and Engineering (UBMK)*. IEEE, sep 2019.

[11] Justus Mattern, Yu Qiao, Elma Kerz, Daniel Wiechmann, and Markus Strohmaier. FANG-COVID: A new large-scale benchmark dataset for fake news detection in German. In *Proceedings of the Fourth Workshop on Fact Extraction and VERification (FEVER)*, pages 78–91, Dominican Republic, November 2021. Association for Computational Linguistics.

[12] Natali Ruchansky, Sungyong Seo, and Yan Liu. CSI. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*. ACM, nov 2017.

[13] Limeng Cui, Kai Shu, Suhang Wang, Dongwon Lee, and Huan Liu. defend: A system for explainable fake news detection. In *Proceedings of the 28th ACM international conference on information and knowledge management*, pages 2961–2964, 2019.

[14] Hunt Allcott and Matthew Gentzkow. Social media and fake news in the 2016 election. *Journal of economic perspectives*, 31(2):211–36, 2017.

[15] Dominic Spohr. Fake news and ideological polarization: Filter bubbles and selective exposure on social media. *Business information review*, 34(3):150–160, 2017.

[16] Dominic DiFranzo and Kristine Gloria-Garcia. Filter bubbles and fake news. *XRDS: Crossroads, The ACM Magazine for Students*, 23(3):32–35, 2017.

[17] Gabor Hulko. Fake news and social media: Regulation and administrative measures in slovakia. *In Medias Res*, page 297, 2021.

[18] Zhong-Qiu Zhao, Peng Zheng, Shou-tao Xu, and Xindong Wu. Object detection with deep learning: A review. *IEEE transactions on neural networks and learning systems*, 30(11):3212–3232, 2019.

[19] Rowan Zellers, Ari Holtzman, Hannah Rashkin, Yonatan Bisk, Ali Farhadi, Franziska Roesner, and Yejin Choi. Defending against neural fake news. *Advances in neural information processing systems*, 32, 2019.

[20] David Ifeoluwa Adelani, Haotian Mai, Fuming Fang, Huy H Nguyen, Junichi Yamagishi, and Isao Echizen. Generating sentiment-preserving fake online reviews using neural language models and their human-and machine-based detection. In *International Conference on Advanced Information Networking and Applications*, pages 1341–1354. Springer, 2020.

[21] Munazza Zaib, Quan Z Sheng, and Wei Emma Zhang. A short survey of pre-trained language models for conversational ai-a new age in nlp. In *Proceedings of the Australasian Computer Science Week Multiconference*, pages 1–4, 2020.

[22] Jose Camacho-Collados and Mohammad Taher Pilehvar. On the role of text preprocessing in neural network architectures: An evaluation study on text categorization and sentiment analysis. *arXiv preprint arXiv:1707.01780*, 2017.

[23] Vimala Balakrishnan and Ethel Lloyd-Yemoh. Stemming and lemmatization: A comparison of retrieval performances. 2014.

[24] Muhammad Yaseen Khan, Abdul Qayoom, Muhammad Suffian Nizami, Muhammad Shoaib Siddiqui, Shaukat Wasi, and Syed Muhammad Khaliq-ur-Rahman Raazi. Automated prediction of good dictionary examples (gdex): A comprehensive experiment with distant supervision, machine learning, and word embedding-based deep learning techniques. *Complexity*, 2021, 2021.

[25] Jong-Yeol Yoo and Dongmin Yang. Classification scheme of unstructured text document using tf-idf and naive bayes classifier. *Advanced Science and Technology Letters*, 111(50):263–266, 2015.

[26] Christopher D Manning. *Introduction to information retrieval*. Syngress Publishing„ 2008.

[27] Deepak Kumar Sharma, Mayukh Chatterjee, Gurmehak Kaur, and Suchitra Vavilala. 3 - deep learning applications for disease diagnosis. In Deepak Gupta, Utku Kose, Ashish Khanna, and Valentina Emilia Balas, editors, *Deep Learning for Medical Applications with Unique Data*, pages 31–51. Academic Press, 2022.

[28] Yutaka Sasaki et al. The truth of the f-measure. *Teach tutor mater*, 1(5):1–5, 2007.

[29] Mark H Zweig and Gregory Campbell. Receiver-operating characteristic (roc) plots: a fundamental evaluation tool in clinical medicine. *Clinical chemistry*, 39(4):561–577, 1993.

[30] James A Hanley and Barbara J McNeil. The meaning and use of the area under a receiver operating characteristic (roc) curve. *Radiology*, 143(1):29–36, 1982.

[31] tutorialandexample. Decision trees in machine learning, 2019.

[32] Klára Komprdová et al. *Rozhodovací stromy a lesy*. Akademické nakladatelství CERM, 2012.

[33] Tom Dietterich. Overfitting and undercomputing in machine learning. *ACM computing surveys (CSUR)*, 27(3):326–327, 1995.

[34] Tin Kam Ho. Recognition of handwritten digits by combining independent learning vector quantizations. In *Proceedings of 2nd International Conference on Document Analysis and Recognition (ICDAR'93)*, pages 818–821. IEEE, 1993.

[35] Tin Kam Ho. Random decision forests. In *Proceedings of 3rd international conference on document analysis and recognition*, volume 1, pages 278–282. IEEE, 1995.

[36] wikipedia. Random forest, 2023.

[37] Naomi S Altman. An introduction to kernel and nearest-neighbor nonparametric regression. *The American Statistician*, 46(3):175–185, 1992.

[38] datacamp. Knn classification tutorial using scikit-learn, 2013.

[39] Jan Švehla. Za nového člověka! sociální experiment v poválečném československu na příkladu emila zátopka. 2021.

[40] Robi Polikar. Ensemble based systems in decision making. *IEEE Circuits and systems magazine*, 6(3):21–45, 2006.

[41] Leo Breiman. Bagging predictors. *Machine learning*, 24(2):123–140, 1996.

[42] Robert E Schapire. The strength of weak learnability. *Machine learning*, 5(2):197–227, 1990.

[43] Ing. Juraj Muráň. Úvod do neurónových sietí - ako fungujú neurónové siete, 2013.

[44] Simon Haykin. *Neural networks and learning machines, 3/E*. Pearson Education India, 2009.

[45] George Cybenko. Approximations by superpositions of a sigmoidal function. *Mathematics of Control, Signals and Systems*, 2:183–192, 1989.

[46] Frank Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386, 1958.

[47] Meriem Bahi and Mohamed Batouche. Deep learning for ligand-based virtual screening in drug discovery. pages 1–5, 10 2018.

[48] Pranoy Radhakrishnan. What are hyperparameters? and how to tune the hyperparameters in a deep neural network. *Towards Data Science*, 18, 2017.

[49] David H Hubel and Torsten N Wiesel. Receptive fields and functional architecture of monkey striate cortex. *The Journal of physiology*, 195(1):215–243, 1968.

[50] Yann LeCun, Yoshua Bengio, et al. Convolutional networks for images, speech, and time series. *The handbook of brain theory and neural networks*, 3361(10):1995, 1995.

[51] Wen-tau Yih, Kristina Toutanova, John C Platt, and Christopher Meek. Learning discriminative projections for text similarity measures. In *Proceedings of the fifteenth conference on computational natural language learning*, pages 247–256, 2011.

[52] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[53] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[54] Leila Kalkhoran, Shima Tabibian, and Elaheh Homayounvala. Detecting persian speaker-independent voice commands based on lstm and ontology in communicating with the smart home appliances. *Artificial Intelligence Review*, 11 2022.

[55] Ming Liang and Xiaolin Hu. Recurrent convolutional neural network for object recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3367–3375, 2015.

[56] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[57] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

[58] Jay Alammar. The illustrated transformer. *The Illustrated Transformer–Jay Alammar–Visualizing Machine Learning One Concept at a Time*, 27, 2018.

[59] Jeffrey Pennington, Richard Socher, and Christopher D Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543, 2014.

[60] Amirhossein Kazemnejad. Transformer architecture: The positional encoding. *Kazemnejad's blog*, 2019.

[61] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[62] Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*, 2016.

[63] Radford Alec, Narasimhan Karthik, Salimans Tim, and Sutskever Ilya. Improving language understanding with unsupervised learning. *Citado*, 17:1–12, 2018.

[64] Abhilash Jain, Aku Ruohe, Stig-Arne Grönroos, and Mikko Kurimo. Finnish language modeling with deep transformer models. *arXiv preprint arXiv:2003.11562*, 2020.

[65] Kevin Clark, Urvashi Khandelwal, Omer Levy, and Christopher D. Manning. What does bert look at? an analysis of bert's attention, 2019.

[66] Petr Zelina. Pretraining and evaluation of czech albert language model.

[67] Patrick Lewis, Barlas Oğuz, Ruty Rinott, Sebastian Riedel, and Holger Schwenk. Mlqa: Evaluating cross-lingual extractive question answering, 2019.

[68] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach, 2019.

[69] Matúš Pikuliak, Štefan Grivalský, Martin Konôpka, Miroslav Blšták, Martin Tamajka, Viktor Bachratý, Marián Šimko, Pavol Balážik, Michal Trnka, and Filip Uhlárik. Slovakbert: Slovak masked language model, 2021.

[70] Rico Sennrich, Barry Haddow, and Alexandra Birch. Neural machine translation of rare words with subword units. *arXiv preprint arXiv:1508.07909*, 2015.

Zoznam obrázkov

Zoznam tabuliek

A. Prílohy

A.1 Technická špecifikácia použitých zariadení

V tejto práci sa na prípravu dát a vykonanie experimentov využívala platforma Google Colab, ktorá poskytuje cloudové výpočtové prostredie. Na výpočty bola využitá grafická karta NVIDIA A100-SXM s dostupnou operačnou pamäťou (RAM) 89,6 gigabajtov. Presné technické špecifikácie grafickej karty sú uvedené na obrázku A.1.

```
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 525.85.12    Driver Version: 525.85.12    CUDA Version: 12.0     |
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|                               |                      |               MIG M. |
|===============================+======================+======================|
|   0  NVIDIA A100-SXM...  Off  | 00000000:00:04.0 Off |                    0 |
| N/A   33C    P0    46W / 400W |      0MiB / 40960MiB |      0%      Default |
|                               |                      |             Disabled |
+-------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+
| Processes:                                                                  |
|  GPU   GI   CI        PID   Type   Process name                  GPU Memory |
|        ID   ID                                                   Usage      |
|=============================================================================|
|  No running processes found                                                 |
+-----------------------------------------------------------------------------+
```

Obr. A.1: Google Colab výpočtová jednotka.

A.2 Použité technológie pri implementácii

V tejto prílohe sa budeme zoberať popisom technológií, ktoré sme použili pri implementácií tejto diplomovej práce, ktorá sa zaoberá detekciou falošných správ. Uvedieme niekoľko nástrojov a technológií, ktoré sme použili, aby sme zabezpečili efektívne a presné spracovanie a analýzu slovenských textov.

Jedným z týchto nástrojov bol Jupyter Notebook, ktorý poskytol interaktívne programovacie prostredie pre jazyk Python 3.9.0. Python sme si vybrali pre jeho vysokú popularitu, jednoduchosť použitia a široké možnosti knižníc a nástrojov. Okrem toho sme využili aj Google Colab, cloudové vývojové prostredie s výkonnými výpočtovými zdrojmi, ktoré nám umožnilo efektívne rýchlejšie iterovanie pri vývoji.

Pri implementácii sme využili aj množstvo knižníc jazyka Python, ktoré sú uvedené v nasledjúcej časti:

1. Na spracovanie a čistenie dát sa použili knižnice:
    - ‘*os*‘ na manipuláciu s adresárovým systémom a načítanie dát z rôznych formátov (napr. .csv, .txt, .json)

- ‘*re*‘ na spracovanie a extrakciu informácií z textov
- ‘*string*‘ na prácu s reťazcami a znakmi
- ‘*nltk*‘ a ‘*simplelema*‘ na tokenizáciu, odstránenie stop slov a stemming slov
- ‘*gensim*‘ na vytvorenie Word2Vec pre reprezentáciu slov v texte

2. Vektorizácia textu sa udiala pomocou knižníc:
   - ‘*sklearn*‘ na vytvorenie vektorových reprezentácií textov
   - ‘*transformers*‘ na vytvorenie vektorových reprezentácií textov pomocou predtrénovaných jazykových modelov (napr. BERT, RoBERTa, SlovakBERT)
3. Trénovanie modelov strojového učenia:
   - ‘*sklearn*‘ na trénovanie klasifikačných modelov (napr. DecisionTreeClassifier, RandomForestClassifier, AdaBoostClassifier, GradientBoostingClassifier, SVC, LinearSVC, MLPClassifier, MultinomialNB, KNeighborsClassifier, XGBClassifier)
   - ‘*tensorflow*‘ a ‘*keras*‘ na vytvorenie hlbokých neurónových sieti
   - ‘*transformers*‘ pomocou, ktorého boli vytvorené veľké predtrénované jazykové modely (napr. BERT, RoBERTa, SlovakBERT)
   - ‘*hyperopt*‘ na optimalizáciu hyperparametrov modelov
   - ‘*Callback*‘ na monitorovanie a zlepšovanie výkonnosti modelov počas trénovania
4. Vyhodnocovanie a vizualizácia výsledkov:
   - Použitie knižnice ‘*sklearn*‘ na vyhodnocovanie výkonnosti modelov pomocou metrík ako presnosť, recall, F1-skóre a ROC krivka
   - Knižnica ‘*matplotlib*‘ bola použitá na vizualizáciu dát a výsledkov
   - Použitie knižníc ‘*pandas*‘ a ‘*numpy*‘ na spracovanie a manipuláciu so vstupnými dátami
   - ‘*argparse*‘ pre preprezentácie parametrov výpočtov
   - ‘*pickle*‘ na uloženie modelov strojového učenia a dátových sád
5. Preklad textov do rôznych jazykov za použitia knižnice:
   - ‘*googletrans*‘ vo verzii 3.1.0*a*0

Celková technická špecifikácia zahŕňa viacero knižníc na spracovanie, vektorizáciu a trénovanie klasifikačných modelov, a zároveň na vizualizáciu ich výsledkov. Použitie týchto knižníc umožnilo efektívne riešenie problému detekcie falošných správ v tejto práci.