VŠB TECHNICKÁ UNIVERZITA OSTRAVA
FAKULTA ELEKTROTECHNIKY A INFORMATIKY

Identifikace autorů v textových dokumentech

Identification of Authors in Text Documents

Bc. Sabína Mackovčáková

Diplomová práce

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

Ostrava, 2024

VŠB TECHNICKÁ UNIVERZITA OSTRAVA | FAKULTA ELEKTROTECHNIKY A INFORMATIKY

Č.j.VSB/24/011472
vsboee240031B9

Zadání diplomové práce

Student: **Bc. Sabína Mackovčáková**

Studijní program: N0613A140034 Informatika

Téma: Identifikace autorů v textových dokumentech
Identification of Authors in Text Documents

Jazyk vypracování: slovenština

Zásady pro vypracování:

Cílem práce bude prostudovat použití strojového učení, neuronových sítích a hlubokých neuronových sítí pro zpracování textu - konkrétně pro určování autorství textů. Cílem bude navrhnout a otestovat metody identifikace autorství textových dokumentů.

Práce bude obsahovat:
1. Přehled metod využitelných pro klasifikaci v textových datech.
2. Detailní popis navržených metod.
3. Definování experimentů a příprava zvolených metod.
4. Ověření fungování navržených metod a porovnání výsledků těchto metod pro více datových sad.

Seznam doporučené odborné literatury:

[1] Torres-Moreno, Juan-Manuel, ed. Automatic text summarization. John Wiley & Sons, 2014.
[2] Mani, Inderjeet, and Mark T. Maybury. Advances in automatic text summarization. MIT press, 1999.
[3] Chris Manning and Hinrich Schütze, Foundations of Statistical Natural Language Processing, MIT Press. Cambridge, MA: May 1999
[4] Dan Jurafsky: Speech and Language Processing, Pearson Education, 2014
[5] Bing Liu: Sentiment Analysis: Mining Opinions, Sentiments, and Emotions, Cambridge University Press; 1 edition (June 4, 2015)
[6] Charu C. Aggarwal: Machine Learning for Text, Springer, 2018.
[7] VASWANI, Ashish, Noam SHAZEER, Niki PARMAR, Jakob USZKOREIT, Llion JONES, Aidan N. GOMEZ, Łukasz KAISER a Illia POLOSUKHIN. Attention is all you need. In: Advances in neural information processing systems. 2017, s. 5998-6008.
[8] ROTHMAN, Denis. Transformers for natural language processing: build innovative deep neural network architectures for NLP with Python, PyTorch, TensorFlow, BERT, RoBERTa, and more. Birmingham: Packt Publishing, 2021. Expert insight. ISBN 978-1800565791.
[9] Rowel Atienza, Advanced Deep Learning with TensorFlow 2 and Keras: Apply DL, GANs, VAEs, deep RL, unsupervised learning, object detection and segmentation, and more. 2nd edition. Packt Publishing, 2020. ISBN 978-1800568273.

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

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

IČ: 61989100
DIČ: CZ61989100

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

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

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

Datum zadání: 01.09.2023

Datum odevzdání: 30.04.2024

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

V IS EDISON zadáno: 16.02.2024 13:36:40

Abstrakt

Práce se zabývá identifikací autorů v textových dokumentech. Cílem práce je otestovat modely strojového učení, neuronových sítí a hlubokých neuronových sítí a jejich vhodnost pro úlohy zpracování přirozeného jazyka a identifikace autorů v textových dokumentech.

Teoretická část pojednává o přípravě dat před jejich použitím jako vstupu do modelů. Pojednává o technikách, které se používají pro předzpracování, dále jsou diskutovány formy vektorizace a vkládání slov. Jsou zde popsány jednotlivé modely z oblasti strojového učení, neuronových sítí a transformer modelů.

V praktické části bylo provedeno několik experimentů, které jsou diskutovány a vyhodnoceny na základě 4 metrik - accuracy, recall, precision, f1 score. Byly testovány vyvážené a nevyvážené datové sady mezi sebou, několik typů vektorizací a nastavení jejich parametrů a modely byly přizpůsobeny tak, aby dosáhly co nejvyšší přesnosti.

Kľúčové slová

identifikace autorství, zpracování přirozeného jazyka, zpracování textu, strojové učení, neuronové sítě, konvoluční neuronové sítě, rekurentní neuronové sítě, architektura Transformer, Bert, DistilBert, Electra

Abstract

The thesis deals with the identification of authors in text documents. The aim of the work is to test machine learning, neural network and deep neural network models and their suitability for natural language processing and author identification tasks in text documents.

The theoretical part discusses the preparation of data before using it as input to the models. It discusses the techniques that are used for preprocessing, and also discusses forms of vectorization and word embedding. Individual models from the fields of machine learning, neural networks and transformer models are described.

In the practical part, several experiments have been conducted and are discussed and evaluated based on 4 metrics - accuracy, recall, precision, f1 score. Balanced and unbalanced datasets were tested against each other, several types of vectorizations and their parameter settings were tested and models were adjusted to achieve the highest accuracy.

Keywords

authorship identification, natural language processing, text processing, machine learning, neural networks, convolutional neural networks, recurrent neural networks, transformer architecture, Bert, DistilBert, Electra

Poďakovanie

Rada by som na tomto miesta chcela poďakovať vedúcemu mojej diplomovej práce, ktorým bol prof. Ing. Jan Platoš, Ph.D. za ochotu pri poskytovaní konzultácií a za odborné rady. Tiež by som chcela poďakovať mojej rodine, ktorá mi bola po celý čas tvorby tejto práce psychickou oporou.

Obsah

Zoznam použitých skratiek a symbolov

NLP	–	Natural Language Processing (Zpracovanie prirodzeného jazyka)
DNN	–	Dense neural network
CNN	–	Convolutional neural network
RNN	–	Recurrent neural network
LSTM	–	Long short-term memory
TF-IDF	–	Term Frequency-Inverse document frequency
BoW	–	Bag of Words
CBOW	–	Continuous Bag of Words
GloVe	–	Global Vectors for Word Representation
BERT	–	Bidirectional Encoder Representations of Transformers
Electra	–	Efficiently Learning an Encoder that Classifies Token Replacements Accurately
RTD	–	Replaced Token Detection
MLM	–	Masked Language Modeling
NSP	–	Next Sentence Prediction
HTML	–	HyperText Markup Language
URL	–	Uniform Resource Locator
TXT	–	Text File Document
CSV	–	Comma-Separated Values

Zoznam obrázkov

Zoznam tabuliek

Kapitola 1

Úvod

V súčasnej digitálnej dobe je potrebné vedieť efektívne spracovať prirodzený jazyk (NLP) a textové dáta. Nachádzame sa v dobe digitálnych veľkých dát, ktoré nie je ľudskou silou možné spracovať. Efektívne spracovávať a analyzovať takéto dáta pomocou technológií dokáže výrazne uľahčiť prácu človeku, jasný príklad je ChatGPT alebo akékoľvek iné umelé inteligencie.

Jednou z oblastí, kde sa NLP využíva je identifikácia autorstva v textových dokumentoch. Ľudia sa už po stáročia snažia nájsť pravých autorov niektorých literálnych diel, kde je autor buď neznámy alebo sa k dielu hlási niekoľko rôznych autorov. Už koncom 19. storočia americký fyzik Mendenhall použil štatistické metódy na kvantifikovanie štýlov písania. Neskôr bol vykonaný významný projekt v tejto oblasti - Federalist Papers project. Bola vykonaná rozsiahla práca na 85 politických dokumentoch (tzv. federalistické dokumenty), ktoré napísali Alexander Hamilton, John Jay a James Madiso, ale pôvodne boli publikované anonymne, pretože autori boli neznámi. Najskôr bol definovaný súbor často používaných stop-slov a pomocou štatistickej analýzy, frekvencií stop slov a obsahových slov, boli získané výsledky a teda pravdepodovný autor pre daný dokument. Aj vďaka tomuto projektu a jeho ukážke praktického využitia identifikácie autorstva sa zvýšil záujem o ďalší výskum v tejto oblasti.

Identifikácia autorstva sa teda vykonáva pomocou štatistických a výpočtových metód a určuje najpravdepodobnejšieho autora daného textu. Predpokladá sa, že každý autor má určité charakteristické črty písania, ktoré sú odlišné od ostatných autorov. Môže ísť napríklad o frekvenciu určitých slov, špecifické slová, dĺžka slov a dĺžka viet, počet slovies vo vete, časté používanie niektorých citosloviec a podobne. Túto úlohu môžeme považovať za klasifikačnú úlohu, kde autori sú jednotlivé triedy a úlohou modelu je ich správne priradiť k textom. Dnes sa dá identifikácia autorstva využit napríklad v oblasti forenznej analýzy pri identifikácií páchateľa na základe písaného dokumentu. V akademickom prostredí na detekciu plagiátorstva v študentských prácach. Stále sa využíva aj pri identifikovaní autorov v literálnych dielach, ktoré autora postrádajú.

Samozrejme s rastom technológií sa práca pri identifikovaní autorstva uľahčila a momentálne sme schopní na tento účel vytvárať veľmi efektívne a presné modely. V práci bude najskôr predstavený

spôsob spracovávania dát. Tie musia byť najskôr upravené tak, aby s nimi vedel počítač lepšie pracovať, čo sa dosiahne najskôr preprocessingom, teda úpravou textových dát. Následne je potrebné dáta previesť do číselnej podoby, takže budú ukázané rôzne spôsoby vektorizácie. Takéto dáta už môžu byť použité ako vstup do modelov. Postupne práca prejde od starších modelov strojového učenia (Logistická regresia a Naive-Bayes), cez neurónové siete (DNN, CNN, RNN) po najnovšie modely na spracovávania prirodzeného jazyka - Transformer.

Ďalej budú predstavené použité technológie pri implementácií tejto práce a samotná implementácia. V rámci nej budú otestované modely, ktoré boli opísané v teoretickej časti. Modely budú vytvárané a upravované s cieľom dosiahnúť čo najväčšiu prenosť. Testovať sa budú na rôznych datasetoch a vektorizáciach za cieľom nájsť čo najlepšiu kombináciu a odhaliť veci, ktoré najviac vedia ovplyvniť presnosť. Všetky experimenty a ich výsledky budú zhrnuté v osobitnej kapitole. Na konci budú zrhnuté poznatky a závery, ktoré mi táto práca poskytla.

Kapitola 2

Zpracovanie textových dát

Textové dáta, ktoré pochádzajú z reálneho sveta sú obvykle neúplne, neštrukturované a zašumené. Je potrebné ich upraviť metódami na predspracovanie dát (preprocessing), tak aby sme dostali vhodné vstupné údaje na ich ďalšie spracovanie. Upravené slová je následne potrebné previesť do číselnej podoby pomocou vektorizácie. Oba tieto procesy budú predstavené v tejto kapitole.

2.1 Zpracovanie prirodzeného jazyka

Zpracovanie prirodzeného jazyka (Natural Language Processing - NLP) je obor umelej inteligencie v spojení s informatikou a lingvistikou, ktorý sa zaoberá interakciou medzi ľudským jazykom a počítačom. Cieľom NLP je spracovať ľudský jazyk tak, aby bol zrozumiteľný pre počítače. Používa rôzne výpočetné techniky na analýzu, porozumenie a následne aj vytvárenie prirodzeného jazyka. To sa využíva v mnohých odvetviach, napríklad:

- **Preklad textu**: pri preklade textu je dôležité správne pochopenie kontextu a významu danej vety, pretože prekladanie štýlom slovo po slove by mohlo byť veľmi nepresné (DeepL, Google Translator).
- **Virutálny agent / chatbot** : snažia sa rozpoznať a pochopit ľudskú reč tak, aby mohli korektne odpovedať na otázku alebo vykonať požadovanú funkciu (Siri, Alexa).
- **Analýza sentimentu**: analyzovanie textu napríklad v recenziách alebo príspevkoch na sociálnych sieťach, na základe čoho je možné zistiť postoj a pocity užívateľov z daného produktu alebo príspevku.
- **Detekcia spamu**: spamy majú veľa charakteristickýh čŕt, ktoré sa v nich pravidelne opakujú a cieľom NLP je ich zachytenie a označenie za spam (napríklad e-mailu). Ide o črty ako finančné termíny, nesprávna gramatika a skomolené názvy spoločností alebo prílišné naliehanie a vyhráženie.

- **Sumarizácia textu**: je potrebné vystihnúť hlavné časti rozsiahlého textu a interpretovať ich tak, aby mal čitateľ predstavu, o čom daný text je a aká je jeho hlavná myšlienka [1].

2.2 Preprocessing

Textové dáta z prirodzeného jazyka obsahujú šum, nadbytočné, nepodstatné a chybné informácie, čo môže komplikovať ich následné spracovanie. Aby mohli algoritmy strojového učenia správne s dátami pracovať a poskytnúť presné výsledky, je nevyhnutné dáta najskôr upraviť. Tento proces sa nazýva preprocessing (predspracovanie). Počas neho sa prevedú rôzne techniky na čistenie, úpravu a zmenšenie dátovej sady. Výsledné dáta sú menšie, čo zabezbečí zvýšenie efektívnosti a zníženie časovej zložitosti modelov. Výber konkrétnych techník na čistenie dát bude závisieť od cieľa spracovania údajov. Niektoré bežne používané techniky budú predstavenehé v tejto podkapitole [2].

2.2.1 Tokenizácia

Tokenizácia je proces rozdeľovania textu na menšie zmysluplné jednotky - odstavce, frázy, slová, symboly a podobne, ktoré sa nazývajú tokeny. Spôsob ako tokenizácia funguje sa môže líšiť od konkrétného jazyka. V angličtine sa tokeny oddeľuje zvyčajne na základe medzier, interpunkčných znamienok a špeciálnych znakov. Rozdelenie textu na slová uľahčí ďalšie spracovanie textu. Tokenizácia sa zvyčajne vykonáva ako jeden z prvých krokov preprocessingu.

2.2.2 Prevod na malé písmena

Aby bola zabezpečená konzistencia a eliminovali sa duplicity slov, ktoré sú identické, len jedno z nich začína veľkým písmenom (napr. je na začiatku vety), zvyknú sa v rámci predspracovania previesť všetky písmená na malé. Niekedy však môže prevod zmeniť význam slova, napr. anglické slovo rose (ruža) a Rose (meno). Na minimalizovanie chýb možno zaviesť určité pravidlá. Napríklad, že sa prevedú len slová na začiatku vety. To ale tiež nezabezpečí úplnu presnosť, pokiaľ bude na začiatku vety meno Rose, prevedie sa na malé rose a zmení svoj význam [2].

2.2.3 Odstránenie nepotrebných častí

Niektoré znaky nenesú v sebe žiadnu informáciu, ktorá by mohla ovyplivniť význam textu, naopak môžu spôsobiť šum a nesprávnu interpretáciu textu. Môže nastať aj problém pri delení textu na tokeny, kde môžu byť znaky zaradené ako samostatné slová. Takéto znaky sa v preprocessingu obvykle odstraňujú. Sú ponecháne v prípade, že nesú potrebnú informáciu pre cieľ analýzy. Príklady znakov, ktoré sa často odstráňujú:

- **Interpunkcia**: znaky ako bodka, otáznik, pomlčka alebo výkričnik.
- **Numerické znaky**: číselné hodnoty, môžu sa odstrániť alebo nahradiť slovným vyjadrením.

- **Špeciálne znaky**: tu sa nachádzajú rôzne znaky ako #, @, * a iné, ktoré sa často vyskytujú na sociálnych sieťach. Nie je vhodné ich odstraňovať, ak je napríklad úlohou detekovať najčastejšie hashtagy na Instagrame (znak #) alebo pokiaľ sa pracuje s e-mailami (znak @).

- **URL adresy a HTML tagy**: neposkytujú žiadne potrebné informácie, slúžia len na korektné zobrazenie v prehliadači.

2.2.4 Stopslová

Stopslová (stopwords) sú slová, ktoré sa v danom jazyku veľmi často vyskytujú, ale nemajú same o sebe žiadny význam. Neovyplvňujú kontext a obsah textu, a ani nemajú žiadny rozlišujúci význam. Napríklad pri klasifikácií článkov, sa bude anglické slovo *the* rovnako často nachádzať v politických aj športových článkoch. Preto nemá zmysel, aby sa takéto slová brali do úvahy. Za stopslová považujeme členy, predložky, spojky a zámená. Každý jazyk má svoj špecifický slovník stopslov, pre angličitnu sú to napríklad {a, an, and, are, as,..}. V každých dátach tiež možno nájsť vlastné stopsslová. Budú to vysokofrekventované slová, ktoré je potrebné identifikovať a nastaviť prah frekvencie. Ak tento prah slovo prekročí, bude považované za stopslovo a bude ostránené. Takýmto odstránením sa výrazne redukuje veľkosť dokumentu [2].

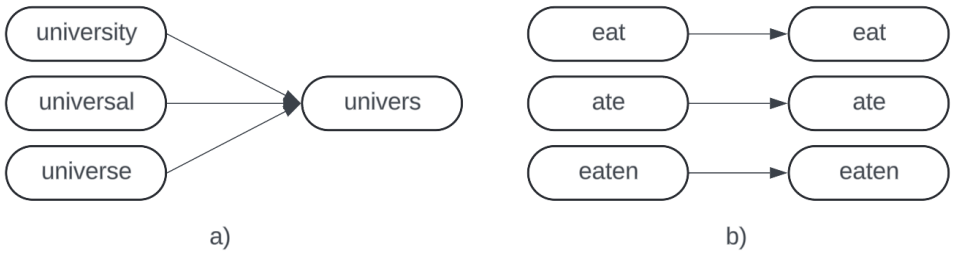
2.2.5 Stemming

Pri stemmingu dochádza k úprave slova na základný - koreňový tvar (stem). V texte sa môže nachádzať slovo v rôznych formách, v jednotnom alebo množnom čísle alebo v rôznych časoch. V týchto prípadoch je vhodné upraviť tieto slová na rovnaký tvar. Algoritmus funguje na základe odstraňovania definovaných predpon alebo prípon, ktoré sú typické pre daný jazyk. Preto výsledný stem nemusí byť vždy reálne slovo, dôležité je, aby sa varianty jedného slova mapovali na rovnaký tvar.

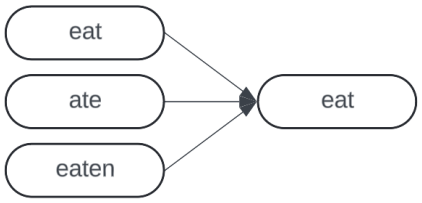
Zníženie počtu unikátnych slov má niekoľko výhod. Môže zabezpečiť lepší výkon modelov, rýchlejší a efektívnejší beh algoritmu. Zjednodušuje to tiež analýzu a porozumenie dátam, čo môže byť výhodné pri analýze sentimentu. Mať zoskupené slová s podobným významom je zasa užitočné pri klasifikácií textu.

Na druhej strane, skracovanie slov len podľa slovníku prípon a predpon môže viesť ku chybám. Môžeme dostať rovnaký základ slova aj zo slov, ktoré majú iný význam (obrázok 2.1a). Naopak, ak slovo počas skloňovania veľmi mení svoj tvar, bude po stemmingu reprezentované rôznymi stemami, aj keď ich význam je rovnaký (obrázok 2.1b). Problém môže nastať aj pri vytváraní algoritmu pri zložitých jazykoch, ktoré nemajú jasné pravidlá na skloňovanie slov [3].

Najznámejšie stemming algoritmy sú Porter Stemmer, Lancaster Stemmer a Snowball Stemmer.



Obr. 2.1: Ukážka stemmingu pomocou algoritmu Porter



Obr. 2.2: Ukážka lemmatizácie

2.2.6 Lemmatizácia

Lemmatizácia je taktiež technika, ktorá upravuje slová na ich koreňový tvar, ktorý sa nazýva lemma. Ide o sofistikovanejší prístup, pretože berie do úvahy kontext daného slova. Používajú sa tu morfologiccké korene slov a výsledná lemma je vždy spisovná. Ukážka je na obrázku 2.2. Táto technika je teda presnejšia, keď je potrebné zachovať v slovách ich význam. Môže ísť napríklad o analýzu sentimentu. Avšak, algoritmus na vykonanie lemmatizácie je časovo náročný, keďže musí prebehnúť morfologická analýza a odvodenie významu slov z rozsiahleho slovníka [3].

2.3 Vektorizácia a Word embeddings

Vektorizáciou je text prevedený na numerickú reprezentáciu (tzv. vektory). Transformuje sa neštruktúrovaný text na štruktúrované číselné údaje, resp. vysokodimenzionálne vektory, ktoré zachytávajú sémantický význam dát v matematickom formáte.

Word embedding môžme voľne preložiť ako vnorené slová. Pri takejto reprezentácií textových údajov sa zachovávava myšlienka, že sémantiku slova možno rezprezentovať z hľadiska jeho kontextu. To je odlišný prístup ako pri klasických modeloch, ktoré reprezentujú každé slovo ako jedinečnú entitu (BoW 2.3.2), bez ohľadu na kontext a sémantiku. Word embeddings reprezentuje slová pomocou viacrozmerného vektorového priestotu, kde priestorová vzdialenosť medzi slovami zodpovedá ich sémentickej alebo jazykovej podobnosti. Pre word embedding existujú 2 hlavné kategórie:

- **Založené na frekencií**: metódy používajú štatistické merania výskytu slova v korpuse.
- **Založené na predpovedaní**: metódy sa učia predpovedať slová zo susedných slov vo vetách. Cieľom je umiestniť slová s podobným kontextom vedľa seba vo vektorovom priestore [4].

V tejto kapitole budú predstavé spôsoby vektorizácie pre strojové učenie a hlboké neurónové siete. Vektorová reprezentácia pre Transformer bude predtavené v samostatnej kapitole 4.

2.3.1 One-hot encoding

One-hot encoding (OHE) je technika, ktorá sa používa na prevod kategorických údajov na číselné hodnoty. Vytvorí binárny stĺpec pre každú kategóriu, kde 0 bude označovať absenciu a 1 prítomnosť kategórie. Binárnou reprezentáciou sa eliminuje nesprávna interpretácia ordinálnych vzťahov, algoritmus nebude môcť pokladať vyššie číslo za lepšiu alebo vhodnejšiu kategóriu. Pri veľkom počte kategórií môže byť OHE neefektívny.

2.3.2 Bag of words

Technika Bag of Words (BoW) je jednou zo základných a najjednoduchších techník reprezentácie textu pomocou čísel. Ignoruje pozíciu a kontext slova vo vete. Pracuje so slovníkom unikátnych slov v dokumente a každému slovu priradí číselnú hodnotu, ktorá predstavuje jeho frekvenciu. Pokiaľ má slovník 100 slov, výsledný vektor bude 100-rozmerný. Jeho i-ty element bude predstavovať frekvenciu i-teho slova zo slovníka. Výsledkom tejto transformácie bude matica, kde stĺpce budú predstavovať unikátne slova naprieč všetkými dokumentami a riadky budú predstavovat jednotlivé dokumenty [5]. Tu je uvedený príklad na 3 dokumentoch:

- **Dokument 1:** NLP je super.
- **Dokument 2:** Dnes je super deň. Dnes študujem.
- **Dokument 3:** Dnes študujem NLP.

Tieto dokumenty obsahujú 6 unikátnych slov. Výsledna matica bude teda mať 6 binárnych stĺpcov a 3 riadky, 1 pre každý dokument.

Dokument	**NLP**	**je**	**super**	**Dnes**	**deň**	**študujem**
Dokument 1	1	1	1	0	0	0
Dokument 2	0	1	1	2	1	1
Dokument 3	1	0	0	1	0	1

Tabuľka 2.1: Bag of Words vektorizácia pre 3 dokumenty

2.3.3 TF - IDF

Technika TF-IDF (Term Frequency-Inverse Document Frequency) sa zameriava na relatívnu dôležitosť slova v dokumente vzhľadom na celý korpus (súbor dokumentov). Výpočet sa skladá z 2 zložiek:

- **TF - Term frequency (frekvencia slova):** meria počet výskytov slova v dokumente, čo sa normalizuje delením počtom všetkých slov v dokumente. Touto normalizáciou sa predchádza skresľovaniu pri rôznych dĺžkach porovnávaných dokumentov. Slovo sa pravdepodobne bude vyskytovať menejkrát v kratších dokumentoch, čo by mohlo viesť k skrelseniu. TF pre slovo t v dokumente d je definovaná ako:

$$\text{TF}(t, d) = \frac{\text{Počet výskytov slova } t \text{ v dokumente } d}{\text{Celkový počet slov v dokumente } d}$$

- **IDF - Inverse Document Frequency (inverzná frekvencia slova):** vyjadruje počet dokumentov v korpuse, ktoré obsahujú dané slovo. Inak povedané, určuje, či je slovo naprieč dokumentami bežné alebo skôr ojedinelé. Menšie číslo (blízke 0) vyjadruje, že slovo je bežné. Väčšia hodnota reprezentuje jedinečné slovo. IDF pre slovo t v korpusu dokumentov D je definované ako:

$$\text{IDF}(t, D) = \log\left(\frac{\text{Celkový počet dokumentov v korpusu } D}{\text{Počet dokumentov obsahujúcich slovo } t + 1}\right)$$

  Pridanie jednotky v menovateli je spôsob, ako predísť deleniu nulou pre slová, ktoré sa vyskytujú vo všetkých dokumentoch.

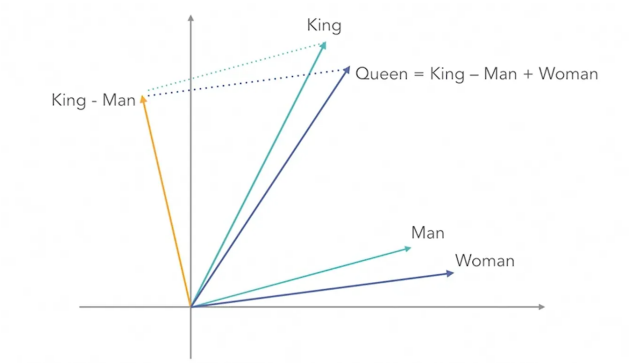
  Konečné TF-IDF skóre pre slovo t v dokumente d v rámci korpusu D sa vypočíta:

$$\text{TF-IDF}(t, d, D) = \text{TF}(t, d) \times \text{IDF}(t, D)$$

Čím vyššie je TF-IDF skóre, tým je slovo charakterickejšie pre daný dokument a je ojedinelé v ostatných dokumentoch korpusu. Táto metóda je výhodná pri identifikovaní kľúčových slov a fráz, ktoré sú unikátne pre jednotlivé dokumenty alebo skupiny dokumentov. Často sa využíva aj pri klasifikáccií textu. Unikátne slová môžu byť rôzne terminológie, geografické údaje, mená, odborné slová a podobne [6].

2.3.4 Word2Vec

Táto metóda bola vyvinutá spoločnosťou Google, reps. jej zamestnancom Tomášom Mikolovom, v roku 2013. Je založená na neurónových sieťach. Pre každé slovo vytvára hustú vektorovú reprezentáciu, pričom zachytáva sémantické a syntaktické vzťahy. Napríklad, vektorové reprezentácie slov *kráľ* a *kráľovná* sú si veľmi podobné a často sa budú vyskytovať v texte spolu. Metóda umožňuje



Obr. 2.3: Algebraické operácie s vektormi [8]

vykonávať algebraické operácie s vektormi, ako je vidieť na obrázku 2.3. Teda ak od vektora slova *kráľ* odčítame vektor *muž* a pripočítame vektor *žena*, dostaneme vektor blízky k vektoru *kráľovná*. Toto ilustruje, ako môže Word2Vec odhadnúť významy slov a ich vzťahy na základe analýzy veľkých textových dát [7].

Pri trénovaní táto sieť používa 2 architektúry - *CBOW* a *SkipGram*.

2.3.4.1 CBOW (Continuous Bag of Words)

Architektúra CBOW je jednoduchá. Skladá sa z 3 hlavných vrstiev: vstupná, skytá a výstupná. Táto architektúra je zobrazená na obrázku 2.4 naľavo. Vstupná vrtsva je reprezentovaná vektorom, kde každý prvok zodpovedá jednému slovu zo slovníka. Veľkosť slovníka teda určuje počet prvkov vo vstupnej vrstve, to isté platí aj pre skrytú a výstupnú vrtsvu. V skrytej vrstve prebieha učenie vnorených vektorov. Pri učení sa využíva algoritmus učenia pod dohľadom. Po natrénovaní výstupná vrtsvá predikuje cieľové slovo. Na konci teda dostaneme vektorovú reprezentáciu slova.

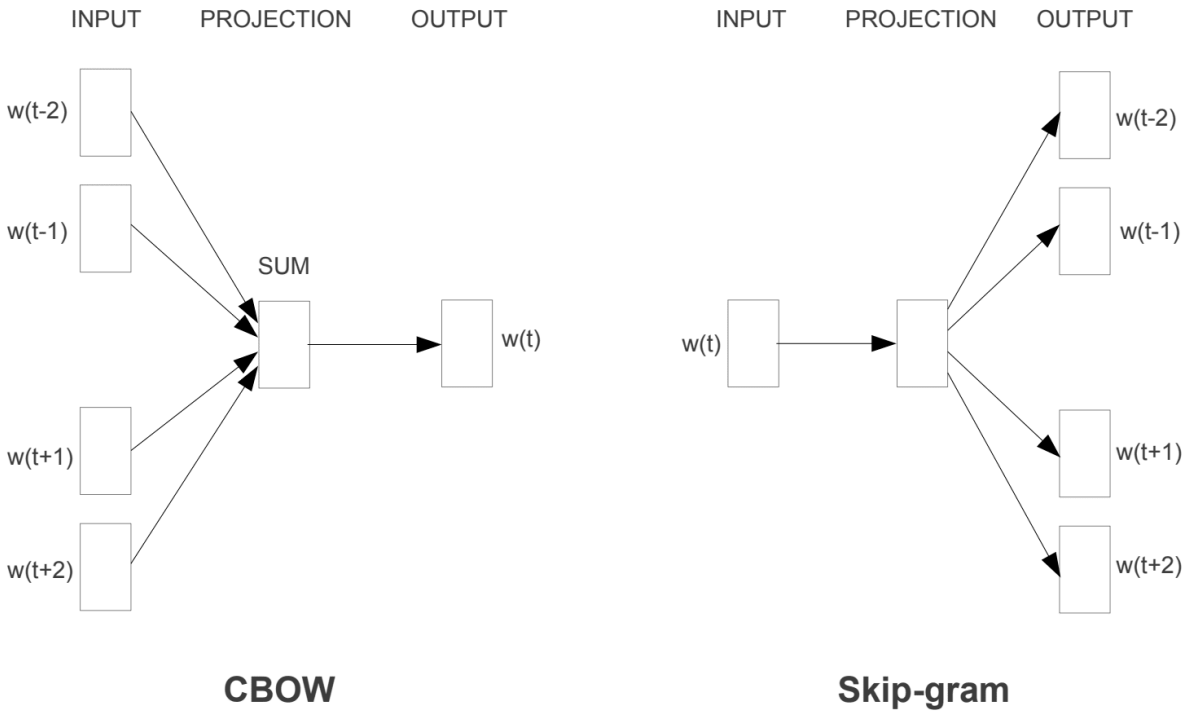
2.3.4.2 Skip-gram

Tento model využíva doprednú neurónovú sieť a predpovedá kontextové slová na základe vstupného slova. Architektúra je podobná ako pri CBOW, skladá sa tiež z 3 vrstiev - vstupná, skrytá a výstupná. Vstupná vrstva tu rezprezentuje cieľové slovo, v skrytej vrstve prebieha učenie a výstupná vrtsva predikuje kontextové slová. Architektúra je zobrazená na obrázku 2.4 napravo.

2.3.4.3 GloVe

GloVe je skratka pre global vectors (globálne vektory). Global odkazuje na globálnu štatistiku korpusu a vektory sú reprezentácie slov. GloVe kombinuje metódy založené na frekvencií s metódami založenými na predpovedaní.

Algoritmus vytvára maticu spoločného výskytu slov, kde každý riadok reprezentuje slovo, stĺpec reprezentuje jeho kontext (iné slovo) a hodnota v bunke predstavuje počet ich spoločných výskytov



Obr. 2.4: Architektúra CBOW a SkipGram [9]

v korpuse. Na naplnenenie tejto matice je potrebný jeden prechod celým korpusom, čo môže byť časovo náročné pre veľké datasety. Po naplnení matice prebehne jej faktorizácia na nižšiu dimenziu. Vzniknutá matica s nižšou dimenziou obsahuje vektorové reprezentácie pre jednotlivé slová. Tieto vektory sú husté (t.j. väčšina ich hodnôt nie je nulová) a každý vektor reprezentuje sémantické a syntaktické vzťahy daného slova vo vzťahu k ostatným slovám v korpusu. Trénovacie iterácie už sú časovo menej náročné, pretože počet nenulových záznamov v matici je v porovnaní s počtom celkových slov nízky [10].

2.3.4.4 FastText

Predchádzajúce techniky (Word2Vec, GloVe) brali ako základnú jednotku jazyka slovo. Avšak, pokiaľ dostaneme nejaký jazyk s bohatou slovnou zásobou alebo s netypickými a nespisovnými slovami, tieto modely ich nespracujú, pretože nebudú súčasťou ich slovníka. Preto bol tímom Facebooku vyvinutý nový nástroj FastText, ktorý by mal zvládnuť aj vzácne slová, resp. slová mimo svojho slovníka.

FastText rozširuje tradičné metódy word embeddings (Word2Vec, GloVe) a generuje embeddings aj pre slová, ktoré sa v tréningových dátach nevyskytujú. Využíva rozkladanie slov na n-gramy. N-gram je časť slova, resp. n po sebe idúcich znakov v slove. Každé slovo je teda reprezentované

ako celok a ako súbor jeho n-gramov. Napríklad ak zvolíme veľkosť n-gramu 2, tak slovo *program* bude reprezentované ako *<program>* a skupina n-gramov: *<pr, ro, og, gr, ra, am>*. Rovnako ako Word2Vec využíva FastText dve architekrúty - CBOW 2.3.4.1 a Skip-gram 2.3.4.2, s tým rozdielom, že okrem trénovania na celých slovách sú trénované aj na n-gramoch [11].

Kapitola 3

State of art

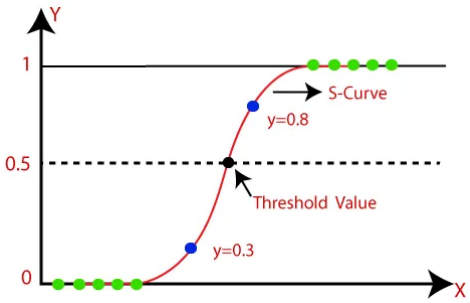
V tejto kapitole budú predstavené súčasne používané modely na zpracovanie prirodzené jazyka a klasifikáciu textu. V kapitole budú predstavené najskôr klasické modely strojového učenia - Logistická regresia a Naive-Bayes. Ďalej budú predstavené neurónové siete, resp. ich architektúry - hustá, rekurentná a konvolučná neurónová sieť.

3.1 Strojové učenie

Strojové učenie je odvetvie umelej inteligencie a informatiky, ktoré sa snaží napodobňovať spôsob ako sa učia ľudia a postupne zlepšovať svoju presnosť. Analyzuje vstupné údaje a hľadá v nich vzory. Podľa US Berkeley [12] proces učenia algoritmu pozostáva z 3 krokov.

1. **Rozhodovací proces:** na základe vstupných údajov, ktoré môžu byť označené alebo neoznačené, vytvorí algoritmus odhad vzoru údajov.
2. **Chybová funkcia:** vyhodnocuje prepoveď modelu. Vykoná porovnanie s existujúcimi známymi prípadmi.
3. **Optimalizácia modelu:** algoritmus zváži chybu a upraví rozhodovací proces tak, aby v ďalšom behu bola chyba menšia.

Metódy strojového učenia môžeme rozdeliť do 2 hlavných kategórií: *supervised (učenie pod dohľadom)* a *unsupervised (učenie bez dozoru)*. Pri učení pod dohľadom dostáva model označené dáta. Dáta teda majú určenú svoju kategóriu a model sa trénuje a prispôsobuje, kým nie je dosiahnutá optimálna presnosť. Medzi takéto modely patria aj Naive-Bayes a Logistická regresia, ktoré budú predstavené v tejto kapitole. Učenie bez dozoru dostáva neoznačené údaje a snaží sa tak zoskupiť dáta s rovnakými vlastnosťami do jeden skupiny. Takto algoritmus objavuje skryté vzory a zoskupenia v dátach. Existuje aj učenie s čiatočným dohľadom, čo je mix medzi prvými dvoma kategóriami. Časť dát je označených a využíva sa pri trénovaní a následne sa podľa toho usmerňuje trénovanie neoznačených dát.



Obr. 3.1: Logisitcká regresia [13]

3.1.1 Logistická regresia

Logistická regresia je štatistická metóda, ktorá sa používa na klasifikáciu a prediktívnu analýzu. Odhaduje pravdepodobnosť výskytu udalosti na základe vstupných premenných, resp. pravdepodobnosť úspechu delenú pravdepodobnosťou zlyhania, ako je ukázané vo vzorci 3.1. Využíva sigmoidnú funkciu, ktorá prevádza každú skutočnú hodnotu na rozsah 0 až 1. Ak je výstup zo sigmoidnej funkcie väčší ako preddefinový prah, model predpovedá, že inštancia patrí do danej triedy. Naopak, ak je výstup menší ako prah, inštancia nepatrí do danej triedy. Ukážka ako to funguje je na obrázku 3.1.

$$\text{logit}(p_i) = \ln\left(\frac{p_i}{1-p_i}\right) \tag{3.1}$$

Na základe typu a počtu tried sa logická regresia rozdeľuje na 3 typy:

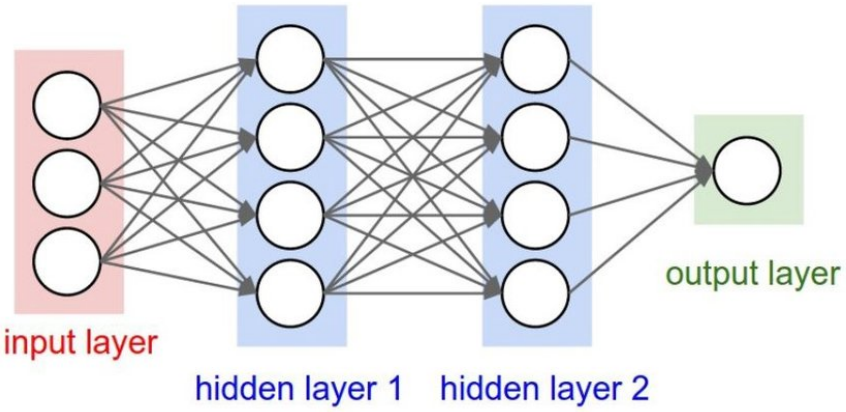
- **Binárna**: má presne 2 triedy (napr. áno, nie).
- **Ordinálna**: má 3 a viac tried, pričom majú definované poradie (napr. školské známky 1-5).
- **Multinomická**: má 3 a viac tried, bez určeného poradia (napr. identifikácia autora) [13].

3.1.2 Naive-Bayes

Naive-bayes je klasifikačný algoritmus založený na princípu Bayesovej vety. Bayesova veta je veta teórie podmienenej pravdepodobnosti, ktorá sa počíta podľa vzorca:

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)} \tag{3.2}$$

- **P(A|B)**: je pravdepodobnosť, že nastane udalosť A za predpokladu, že nastala udalosť B.
- **P(B|A)**: je pravdepodobnosť, že nastane udalosť B za predpokladu, že nastala udalosť A.
- **P(A)**: je predchádzajúca pravdepodobnosť udalosti A.



Obr. 3.2: Architektúra DNN [16]

- **P(B)**: je predchádzajúca pravdepodobnosť udalosti B.

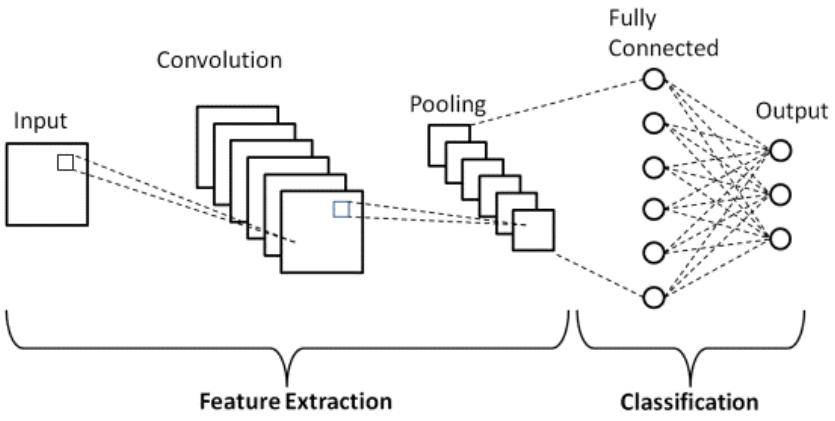
Naive-Bayes predpokladá, že medzi jednotlivými vlastnosťami údajov neexistuje závilosť. Model vypočíta pravdepodobnosti pre každú triedu na základe jej vlastností. Následne bude ako výslednú triedu predikovať tú s najvyššou pravdepodobnosťou.

3.2 Hlboké učenie

Neurónové siete patria k metódam umelej inteligencie, ktoré sa snažia pri učení napodobniť proces, akým sa učí ľudský mozog. Je to súbor malých výpočtových jednotiek, ktoré spracovávajú vstupné hodnoty a generujú výstupné hodnoty. Tieto siete sú inšpirované biologickými neurónmi v mozgu a majú svoje korene v modeli McCulloch-Pitts neurónu z roku 1943. Dnešné moderné neurónové siete, sú skôr sieťami malých výpočtových jednotiek, kde každá jednotka spracováva vstupné vektory a produkuje výstupné hodnoty. Táto architektúra umožňuje neurónovým sieťam učiť sa a adaptovať sa na rôzne druhy dát bez explicitného programovania. Moderné neurónové siete majú často veľmi veľa vrstiev, sú teda *hlboké* a patria medzi modely hlbokého učenia [14].

3.2.1 Plne prepojená NN

Plne prepojená neurónová sieť (Dense Neural Network - DNN) je jednoduchá architektúra, ktorá ma simulovať spôsob, akým ľudský mozog spracováva informácie. Skladá sa z neurónov, ktoré sa počas učenia samé optimalizujú. Neuróny sú husto prepojené, každý neurón je prepojený so všetkými neurónmi z predchádzajúcej vrstvy a zároveň slúži ako vstup pre všetky neuróny z nasledujúcej vsrtvy. Táto architekrúra je zobrazená na obrázku 3.2. Informácie prúdia len jedným smerom, zo vstupnej vrstvy cez jednu alebo viacero skrytých vrstiev do výstupnej vrstvy. To umožňuje zachytiť zložité vzťahy a interakcie medzi vstupnými dátami [15].



Obr. 3.3: Architektúra CNN [17]

3.2.2 Konvolučná NN

Konvolučná neurónová sieť (Convolutional Neural Network - CNN) je podobná DNN, ale bola vyvinutá primárne na analýzu obrázkov. *Konvolučná* v názve odkazuje na matematickú funkciu konvolúcie. Konvolúcia je operácia, ktorá vynásobí dve funkcie a ako výsledok vznikne tretia funkcia. Konkrétne pre obrázky, 2 matice, ktoré reprezentujú obrázky sa vynásobia a výsledok sa používa na extrahovanie prvkov z obrázka.

Architektúra CNN (obrázok 3.3) sa dá rozdeliť na 2 časti hlavné časti. Prvá časť - **extrakcia vlastností** identifikuje rôzne vlastnosti z obrazu. Skladá sa z viacerých konvolučných a združovacích (pooling) vrstiev. Druhá časť - **klasifikácia** je plne prepojená vrstva, ktorá predpovedá triedu obrázku pomocou vlastností získaných pri extrakciií. CNN využíva 3 hlavné vrstvy:

- **Konvolučná vrstva**: extrahuje rôzne vlastnosti zo vstupných obrázkov pomocou matematickej operácie konvolúcie medzi vstupným obrázkom a filtrom určitej veľkosť MxM. Filtrom sa prechádza cez obrázok a posstupne sa vypočítava skalárny súčin s jednotlivými časťami obrázka. Výsledkom je mapa vlastností, ktorá obsahuje informácie o hranách a rohoch obrázka. Táto mapa je následne poskytnutá ďalším vrstvám na učenie.

- **Pooling vrstva**: jej cieľom je znížiť veľkosť mapy vlastností, aby sa znížili výpočtové náklady. Vrstva zjednodušuje vlastnosti extrahované konvolučnou vrstvou a znižuje počet výpočtov pomocou operácií ako max pooling (vyberie sa najväčší prvok) alebo average pooling (vypočíta sa priemerný prvol).

- **Plne prepojená vrtsva**: obsahuje neuróny, váhy a skreslenia a slúži na prepojenie neurónov medzi rôznymi vrstvami. Vstup z predchádzajúcich vrstiev sa sploští a vstúpi do tejto vrstvy, kde sa vykonávajú matematické operácie a začína proces klasifikácie. Plne prepojené vrstvy sú zvyčajne umiestnené pred výstupnou vrstvou a tvoria niekoľko posledných vrstiev celej CNN architektúry [17].

Napriek tomu, že CNN bola pôvodne vyvinutá na prácu s obrázkami, v priebehu času sa ukázalo, že si dokáže dobre poradiť aj so spracovaním prirodzeného jazyka. Ako bolo rozobrané v kapitole X, predtým ako sa ide text spracovávať modelmi strojového učenia je prevedený na číselnú reprezentáciu. Na takýchto údajoch môžu konvolučné vrstvy CNN hľadať určité vzorce a vlastnosti.

Boli aj vykonané štúdie, ktoré skúšali využiť CNN na niečo iné ako prácu s obrázkami. Prvá štúdia [18] nepracovala priamo s textovými dátami, ale súsetredila sa na identifikácia autora zdrojového kódu v rôznych jazykoch ako C++, Java alebo Python. Kód bol prevedený na číselnú reprezentáciu pomocou TF-IDF a word embeddings. Po natrénovaní dosahovali CNN modely presnosť až 96.2% pre 1600 programátorov.

Ďalšia štúdia [19] porovnávala, či je vhodnejšie využiť RNN s LSTM vrstvami alebo CNN na spracovanie prirodzeného jazyka. Výsledok štúdie bol, že neeexistuje 1 ideálny typ neurónovej siete, ktorý vždy podá pri NLP najlepšie výsledky. Vo väčšine prípadov na tom boli o niečo lepšie RNN a LSTM modely, CNN viedlo v úlohe, kde bolo potrebné vybrať z možností správnu odpoveď alebo odpovede pre danú otázku.

3.2.3 Rekuurentná NN

Rekurentné neurónové siete (RNN) boli navrhnuté na spracovávanie sekvenčných údajov. Od predchádzajúcich neurónovyh sietí (DNN, CNN) sa líšia architektúrou, kde sú pridané rekurentné cykly a zavádzajú koncept pamätania si informácie z predchádzajúcich vrstiev. Základné princípy RNN sú:

- **Sekvenčné pripojenia**: príjmajú vstupy sekvenčne jeden po druhom. Každý nový vstup je spracovaný v kontexte informácií získaných z predchádzajúcich vstupov.
- **Rekurentné pripojenia**: vďaka týmto pripojeniam si RNN môžu pamätať predchádzajúce informácie. V každom kroku sa kombinuje aktuálny vstup s tzv. skrytým stavom, ktorý obsahuje informácie získané z predchádzajúcich vstupov.
- **Skrytý stav**: tento stav je dynamicky aktualizovaný na základe nového vstupu a predchádzajúce skrytého stavu v každom časovom kroku.
- **Zdieľané váhy**: všetky váhy, resp. paramatre sú zdieľané v každom časovom kroku. Na spracovanie každého vstupu sa používajú rovnaké váhy, čo zjenodušuje model a zvyšuje efektívnosť modelu [20].

Tieto princípy umožňujú RNN adaptovať sa a reagovať na zmeny v dátových sekvenciách, čo je nemožné pre iné typy neurónových sietí, ktoré nedisponujú pamäťou. Vďaka týmto vlastnostiam majú RNN široké využitie v NLP, rozpoznávaní reči alebo predpovedi časových rád. Problém RNN je v miznúcom gradiente, čo im bráni zvládať dlhodobé závislosti.

3.2.3.1 LSTM

LSTM (Long Short-Term Memory) sú špeciálne typy rekurentných neurónových sietí navrhnuté na učenie dlhodobých závislostí v sekvenčných dátach, teda aj na riešenie problému miznúceho gradientu.

Tieto siete majú tzv. pamäťové bunky, ktoré umožňujú uchovávať informácie po dlhšiu dobu. Pamäťová bunka má 3 komponenty: vstupnú bránu, zabúdaciu bránu a výstupnú bránu. Vstupná brána určuje, ktoré informácie z nového vstupu sa pridajú do bunky. Zabúdacia brána určuje, ktoré informácie sa majú vyradiť. Výstupná brána riadi, aké informácie sú na výstupe z bunky a teda koľko obsahu pamäťovej bunky by sa malo použiť na výpočet skrytého stavu. Tento mechanizmus umožňuje LSTM selektívne uchovávať alebo zahadzovať informácie počas spracovania, čo im umožňuje efektívne sa učiť z dlhodobých dátových vzorov.

3.2.3.2 GRU

GRU (Gated Recurrent Unit) sú zjednodušenou variantou LSTM. Má iba 2 brány - aktualizačnú a resetovaciu. Aktualizačná brána riadi, koľko informácií z predchadzajúceho skrytého stavu sa má zachovať a odovzdať ďalej. Brána resetovania určuje, koľko z minulých informácií sa má zabudnúť. Tento prístup môže riešiť problém miznúceho gradientu, pretože je možné ponechať a odovzdať ďalej aj všetky informácie z minulosti [21].

3.2.3.3 Obojsmerná RNN

Obojsmerná RNN má dve rekuretné skryté vrtvy. Jedna spracováva vstupnú sekvenciu dopredu a druhá dozadu. Počas trénovania sa zhromažďujú informácie v týchto vrstvách a vkladajú sa do poslednej vsrtvy na vytváranie predikcií. Na vytvorenie opakujúcich sa vrstiev sa dá použiť napríklad LSTM alebo GRU.

Kapitola 4

Transformer

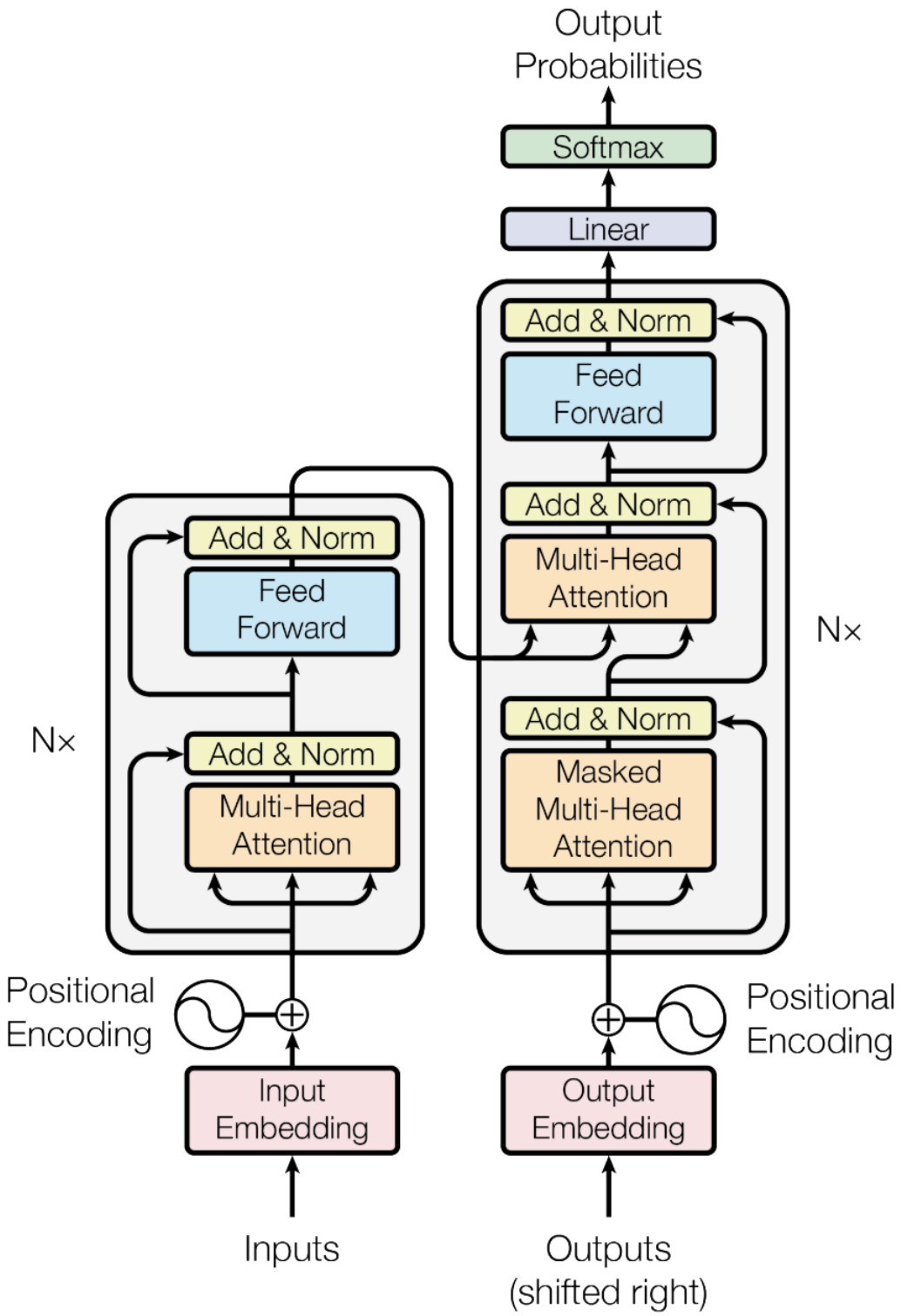
Doterajšie spomínane architektúry (RNN, LSTM, GRU) môžu byť veľmi výkonné pri NLP, no napriek tomu a stále stretávajú s určitými obmedzeniami. Dáta sú spracovávane sekvenčne, prvok po prvku, čo spomaľuje trénovanie. Tiež stále nemajú úplne vyriešený problém miznúceho gradientu. Na riešenie týchto problémov bola spoločnosťou Google vyvinutá nová architekrúra Transformer. Bola predstavená v roku 2017 v práci s názvom ***Attention is All You Need*** [22]. Táto architektúra sa vyhýba rekurencii, namiesto toho sa spoliehha na tzv. attention mechanism (mechanizmus pozornosti). Celá architektúra bude podrobnejšie popísaná v tejto kapitole, spolu s jej využitím v NLP. Pri tvorení tejto kapitoly boli primárne využívané dva zdroje [22] [23].

4.1 Architektúra

Pôvodná architektúra (Obr. 4.1) je zložená z 2 zásobníkov, každý má 6 plne prepojených vrstiev. Zásobník na ľavej strane sa nazýva enkóder, zásobník na pravej strane dekóder. Výstup z vrstvy I bude vstupom vrstvy $I + 1$, pokiaľ sa nedosiahne finálna prepoveď. Označenia Nx pri enkóderi aj dekóderi značí, že sú za sebou naukladané n-krát. Teda výstup jedného enkódra slúži ako vstup pre ďalší enkóder, rovnako to platí aj pre dekódery.

4.1.1 Input embedding

Prvý krok je podobne ako aj pri ostatných neurónových sieťach prevod vstupného textu na vektory, ktoré majú stanovenú dĺžku a obsahujú informácie o význame týchto slov. Každé slovo z vety sa najprv rozdelí na menšie časti (tokeny) a každý token sa potom premení na vektor pomocou naučených vzorcov.



Obr. 4.1: Architektúra Transformer [22]

4.1.2 Positional Encoding

Transformer model neobsahuje rekurziu ani konvolúciu a preto je potrebné inak vyriešiť informácie o relatívnej a absolútnej polohe slov vo vete. Na vyriešenie tohto problému bolo pridané pozičné kódovanie. V architektúre sa vyskytuje dvakrát, pred začiatkom enkódera aj dekódera. Pre každú možnú pozíciu sa vytvorí vektor, vďaka čomu je možné vytvoriť maticu, ktorá bude ukazovať všetky pozície, ktoré môže slovo zaujať. Aby malo každé slovo jedinečnú pozíciu a zároveň bolo aj normalizované na hodnoty [-1, 1], boli v pôvodnom predstavení Tranformeru použité funkcie sinus a cosinus:

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

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

Vstupné vektory z vrstvy input embeddings aj vektory vytvorené pozičným kódovaním majú rovnakú dimenziu, takže ich je možné sčítať.

4.1.3 Enkóder

Každá zo 6 vrstiev enkódera sa skladá z 2 hlavných podvrstiev. *Multi-head attention* mechanizmus a *Plne prepojená dopredná sieť*. Okolo každej podvrstvy je použité residuálne spojenie, ktoré prenesie vstup priamo k výtupu z podvrstvy, potom následuje normalizácia. Teda výstup z každej podvrtvy je definovaný ako:

$$LayerNormalization(x + Sublayer(x)) \tag{4.1}$$

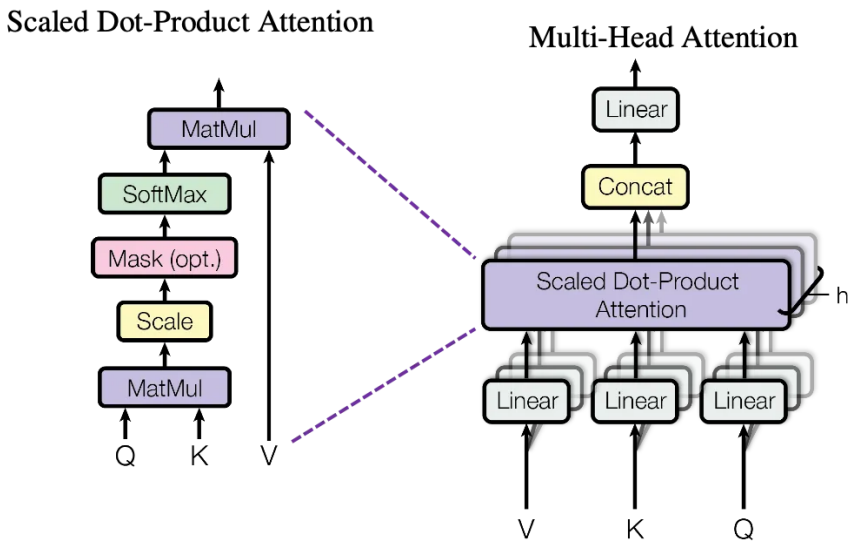
Funkcia Sublayer je implementovaná danou podvrtsvou. Týmto prístupom je zabezpečené zachovanie podstatných informácií (napr. pozičného kódovania). Všetky vrstvy aj podvrstvy vytvárajú výstupy s konštantným rozmerom, v pôvodnej architektúre bola táto dimenzia nastavená na 512, ale dá sa meniť podľa potrieb.

4.1.4 Self-Attention

Self-attention je mechanizmus, ktorý modelu umožňuje vážiť význam jednotlivých častí vstupnej sekvencie. Je to kľúčový prvok Transformerov. Funguje tak, že každý prvok vstupnej sekvencie je prepojený s každým iným prvkom na základe vypočítanej vzájomnej pozornosti alebo dôležitosti. Model tak dokáže identifikovať a zvýrazniť relevantné vzťahy medzi slovami alebo bez ohľadu na ich pôvodné pozície v sekvencii.

4.1.4.1 Výpočet self-attention

Výpočet self-attention sa skladá z viacerých krokov. Ako prvé prebehne lineárna tranformácia vstupných vektorov Z na vektory Q, K a V.



Obr. 4.2: Scaled Dot-Product Attention a Multi-head Attention [22]

- Q(uery): reprezentuje slovo, pre ktoré chceme vypočítať self-attention.
- K(ey): reprezentuje každé slovo v sekvencii a používa sa na porovnanie s query vektorom.
- V(alue): skutočná reprezentácia každého slova v sekvencií.

$$Q = Z * W_Q$$

$$K = Z * W_K$$

$$V = Z * W_V$$

Ďalším krokom je výpočet skóre pozornosti. Skalárnym súčinom Q a K získame skóre, ktoré sa vydelí druhou odmocninou dk (dimenzia key vektora). Následne sa aplikuje funkcia softmax na získanie váh pozornosti, teda hodnoty pozornosti pre každé slovo v sekvencii. Na konci sú tieto váhy vynásobené vektorom V. Tento proces je definovaný rovnicou:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{4.2}$$

Celý tento process a nazýva aj *Scaled Dot-Product Attention* a jeho architektúra je zobrazená na obrázku 4.2 na ľavej strane [24].

Tieto kroky sa budú opakovať h-krát, resp. budú bežať paralelne medzi viacerými hlavami, kde h je počet hláv v multi-head attention mechanizme. Každá hlava sa učí rôzne matice váh, tým sa zameriava na iné vlastnosti v texte. Na dosiahnutie finálneho výstupu z multi-head attention sa

jednotlivé výstupy agregujú, napríklad zreťazením. Obrázok 4.2 napravo ilustruje, ako sa viaceré hlavy používajú súčasne. Konečný výstup je vážený súčet hodnôt.

4.1.5 Plne prepojená dopredná sieť

V doprednej neurónovej sieti dáta prúdia iba smerom dopredu, z jednej vrstvy do druhej. Nie sú tam pridané žiadne slučky ani cykly. Začína sa na vstupnej vrstve, následne dáte prejdú postupne cez všetky skryté vrstvy a skončia vo výstupnej vrstve.

Druhou podvrstvou v každej vrstve enkódera aj dekódera je plne prepojená dopredná sieť. Každá pozícia sa v nej spracováva samostatne a identickým spôsobom. Obsahuje 2 lineárne tranformácie a používa aktivačnú funkciu ReLU.

4.1.6 Dekóder

Architektúra enkódera a dekódera je veľmi podobná. Pri dekóderi ma každá zo 6 vrstiev okrem 2 podvrstiev, ktoré má aj enkóder, ešte 1 podvrstvu *Masked multi-head attention*. V tejto podvrstve sú nasledujúce slová maskované, známe sú len slová po aktuálnu pozíciu. Tým je model nútený predpovedať ďalšie slovo, len s využitím už známych slov. Residuálne prepojenia a normalizácia je rovnaká ako pred enkóder.

4.1.7 Výstup

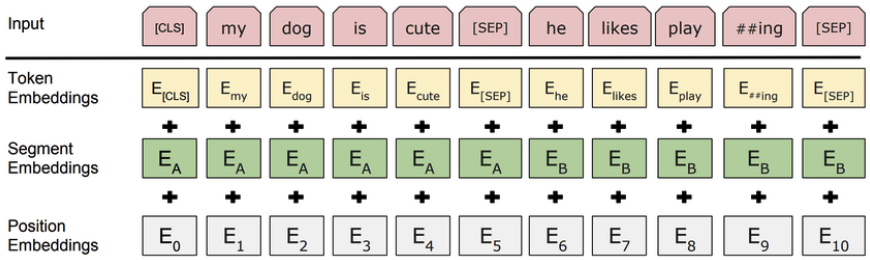
Výstup z dekódera sa pomocou lineárnej vrstvy a softmax funkcie premietne na pravdepodobnosti. Lineárna vrstva vytvorí ďalšie pravdepodobné prvky sekvencie, ktoré funkcia softmax prevedie na pravdepodobný prvok.

4.2 Rozšírenia

Po predstavení Transformera boli vyvinuté ďalšie vylepšenia tejto architektúry, niektoré konkrétne pre NLP. Budú predstavené v tejto kapitole.

4.2.1 Bert

Bert, celým názvom *Bidirectional Encoder Representations from Transformers*, bol predstavený výskumnímmi z Google AI Language [25]. Hlavným prvkom je obojsmerný tréning Transformera, dokáže predpovedať slová na základe predchádzajúcich aj nasledujúcich slov. Bert sa skladá z 2 základných krokov *pretraining* a *fine-tuning*. Bert sa po natrénovaní na dátach z Wikipedie osvedčil v mnohých oblastiach NLP, kde dosahoval veľmi dobre výsledky. Išlo napríklad o odpovedanie na otázky, analýza sentinemntu, generovanie a sumarizácia textu alebo jazykový preklad.



Obr. 4.3: BERT reprezentácia vstupu [25]

4.2.1.1 Architektúra

Bert používa len 1 časť z pôvodnej vrstvy Transformera a to enkóder. Jeho podrobný popis je v kapitole 4.1.3. Vrstvy enkódera u BERT-a sú väčšie ako v pôvodnom Transformer modeli. V pôvodnom článku [25] boli predstavené 2 modely:

- **BERT_base**: 12 enkóder vrstiev za sebou, 768 dimenzií, 12 hláv v multi-head attention, 110 miliónov parametrov.
- **BERT_large**: 24 enkóder vrstiev za sebou, 1024 dimenzií, 16 hláv v multi-head attention, 340 miliónov parametrov.

4.2.1.2 Reprezentácia vstupných dát

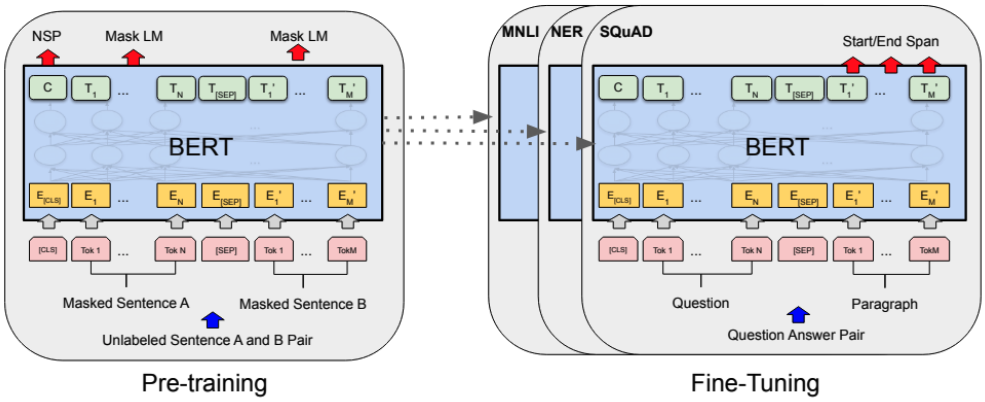
Reprezentácia inputu musí byť jednoznačná a jasne oddeľovať jednu vetu alebo pár viet (napr. otázka - odpoveď). BERT používa embedding WordPiece so slovnou zásobou 30 000 tokenov. Prvý token v každej sekvencií je vždy rovnaký a označuje sa [CLS], čo je skratka pre klasifikačný token. Dvojica viet môže byť zabalená do 1 sekvencie, vtedy treba vety rozlíšiť. Najskôr sa vety rozdelia špeciálnym tokenom [SEP], ktorý označuje koniec každej vety. Potom sa ku každému tokenu pridá naučený embedding, ktorý označuje, či patrí do vety A alebo do vety B (NSP). Reprezentácia pre daný token je vytvorená sčítaním token embeddingu (WordPiece), segmentu (NSP) a pozície, ako je vidieť na obrázku 4.3

4.2.1.3 Pretraining - predtrénovanie

Na predtrénovanie modelu bol použitý korpus BooksCorpus (800 miliónov slov) a anglická Wikipédia (2 500 miliónov slov). Predtrénovanie modelu Bert sa skladá z 2 úloh:

- **MLM - Masked language modeling**

  Obojsmerné trénovanie by každému slovu umožnilo vidieť samého seba a model by mohol jednoducho predpovedať cieľové slovo. Aby bolo možné vykonať efektívne obojsmerné trénovanie, je náhodne maskované určité percento (v pôvodnom článku išlo o 15%) vstupných tokenov,



Obr. 4.4: Bert Pre-trainnig a Fine-Tuning [25]

ktoré sa následne model snaží predpovedať. Týmto spôsobom ale vznikne nesúlad medzi pre-traingom a fine-tuningom, protože vo fine-tuningu sa nezobrazuje maskovaný token [MASK]. Na zmiernenie tohto nesúladu vznikli 3 pravidlá pre vybraté maskované tokeny:

1. Token sa nahradí tokenom [MASK] v 80% prípadoch.
2. Token sa nahradí náhodným slovom zo slovníka v 10% prípadoch.
3. Token sa nenahradí ničím v 10% prípadoch.

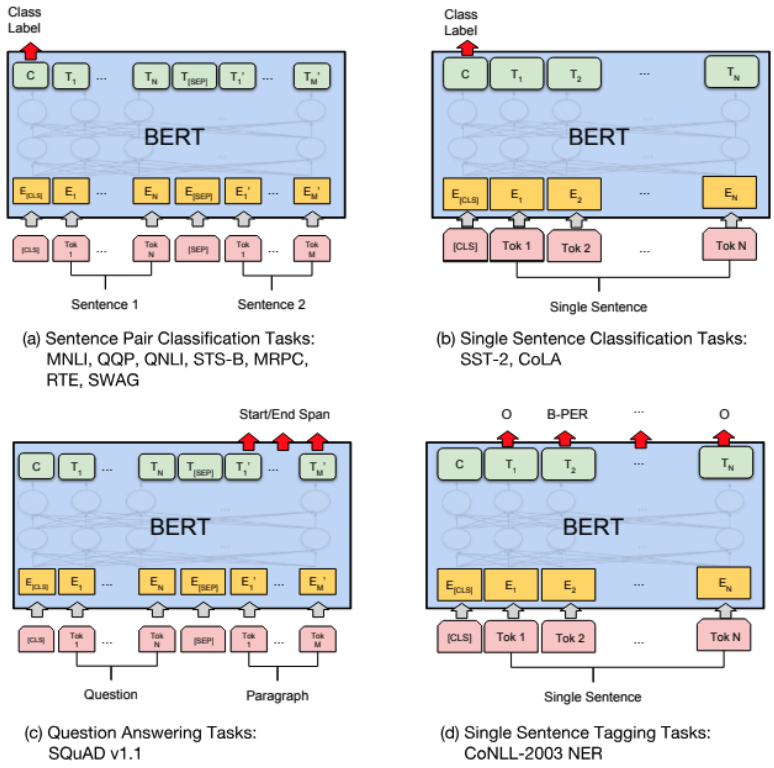
Následne je možné vykonať predikciu maskovaného tokenu. Stratová funkcia Bert-a berie do úvahy len predikciu maskovaných tokenov, nemaskované úplne ignoruje.

- **NSP - Next Sentence Prediction**

  Veľa úloh, na ktoré sa model Bert používa, si vyžaduje pochopenie vzťahu medzi dvoma vetami. Aby model mohol rozumieť väzbám viet, je na to trénovaný už počas úlohy NSP. Pre každý tréningový príklad sú vybraté dve vety, veta A , čo je prvá veta a veta B. V polovoci prípadov je B skutočná nasledujúca veta (označená IsNext) a v druhej polovici je to náhodná veta z korpusu (označená NotNext). Na začiatok vety A je pridaný token [CLS] a na koniec každej vety je pridaný token [SEP]. Tieto pridané tokeny a rozdielenie slov, na slová vety A a slová vety B je zobrazené na obrázku 4.3. Finálny skrytý vektor [CLS] (skrátene len C) sa používa na predikciu ďalšej vety a je zobrazený na obrázku 4.4.

4.2.1.4 Fine-tuning

Fine-tuning je proces prispôsobovania predtrénovaného modelu BERT-a na špecifickú úlohu NLP. Pre každú úlohu je potrebné pripojiť špecifické vstupy a výstupy do modelu BERT a doladiť všetky parametre. Fine-tuning sa vykonáva podľa konkrétnej úlohy, napríklad:



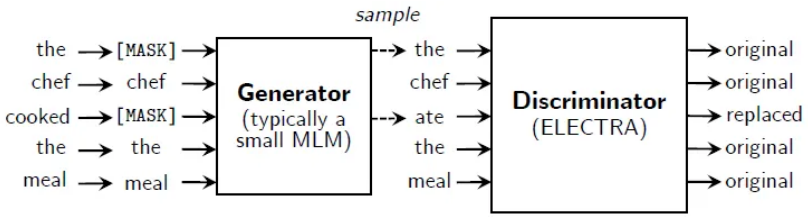
Obr. 4.5: Bert pretrainnig [25]

- **Klasifikácia**: pre klasifikačné úlohy (napr. analýza sentimentu) sa používa reprezentácia [CLS] tokenu, ktorá je smerovaná do výstupnej vrstvy navrhnutej na klasifikáciu.
- **Odpovedanie na otázky**: pri úlohach, kde je potrebné identifikovať odpoveď na danú otázku sa model trénuje na učenie sa identifikovať začiatok a koniec odpovede v texte.
- **Rozpoznávanie pomenovaných entít - NER**: pri úlohach, kde treba identifikovať a označiť rôzne typy entít, sa výstupný vektor každého tokenu privedie do klasifikačnej vrstvy, ktorá predpovedá označenie NER.

Fine-tuning je oproti predtrénovaniu menej náročný na procesor a trvá výrazne kratší čas, pretože model už má všeobecné znalosti jazyka a len sa upravuje na špecifické vzory v nových dátach. Na obrázku 4.5 sú zobrazené mnohé úlohy NLP, na ktoré sa dá BERT využiť.

4.2.2 DistilBERT

DistilBert má názov odvodený od slova destilled (destilovaný), čo znamená, že by mal byť oproti Bert modelu menší, rýchlejší a lacnejší (vzhľadom na GPU). Pri jeho predstavení [26] bolo uvedené, že je o 40% menší, o 60% rýchlejší a zachová až 97% schopnosti porozumieť



Obr. 4.6: Electra diskriminátor a generátor [27]

prirodzenému jazyku oproti pôvodnému modelu. DistilBERT sa destiluje vo veľkých dávkach (až 4 000 inštancií na dávku) pomocou dynamického maskovania a bez cieľa predikcie ďalšej vety. DistilBERT bol trénovaný na rovnakom dátovom korpuse ako BERT. Základné rozdiely medzi modelmi sú:

- **Počet vrstiev**: DistilBert má len 6 transformačných vrstiev, čo je polovica oproti modelu BERT_Base.
- **Počet parametrov**: Vďaka zmenšeniu počtu vrstiev a ďalších optimalizácií má DistilBert o 40% menej parametrov ako Bert, čo znižuje jeho veľkosť a zvyšuje rýchlosť trénovania.
- **Efektívnosť**: Vďaka menej vrstvám a parametrom, je DistilBert rýhlejší a je vhodnejší pre prostredia s obmedzenými vypočtovými zdrojmi.
- **Technika destilácie**: Destilácia je technika kompresie, kde je menší model (DistilBert) trénovaný, aby napodobnil chovanie väčšieho modelu (Bert). Tento proces pomáha zachovať účinnosť pôvodného modelu v kompaktnejšej forme.

Napriek svojej redukovanej veľkosti DistilBert úspešne zvláda veľkú časť schopností Bert-a, dosahuje kvalitné výsledky v širokej škále úloh NLP.

4.2.3 ELECTRA

Pri modeli ELECTRA (PRE-TRAINING TEXT ENCODERS AS DISCRIMINATORS RATHER THAN GENERATORS) sa na predtrénovanie nepoužíva MLM, ale úplne nový prístup *RTD - Replaced Token Detection (detekcia nahradených tokenov)*. Elektra trénuje dve siete - *Generátor* a *Diskriminátor*, každý z nich je zložený z enkóderu Transformera. Generátor nahradí niektoré tokény zo vstupu synteticky generovanými náhradami. To je náhrada techniky MLM z Bert-a. Diskriminátor dostáva ako vstup vety s niektorými slovami nahradenými generátorom. Úlohou diskriminátora je určiť, ktoré slová boli nahradené – čiže diskriminovať medzi pravými a falošnými tokenmi [27]. Tento proces je zobrazený na obrázku 4.6.

Kapitola 5

Použité Technológie

5.1 Tensorflow

TensorFlow je open-source knižnica vytvorená tímom Google Brain a uvedená na trh v roku 2015. V čase písania tejto práce je najnovšia verzia *v2.16.1*. Tensorflow umožňuje vývojarom vytvárať aplikácie najmä pomocou Pythonu, ale aj s C++ a JavaScriptom. Tensorflow umožňuje využívať rôzne modely strojového učenia a neurónových sieti, ktoré sa môžu trénovať na rôzne klasifikácie (textu aj obrázkov) alebo na NLP [28].

5.2 Keras

Keras slúži ako rozhranie pre Tensorflow a je to zároveň knižnica na tvorbu neurónových sietí pomocou jazyka Python. Poskytuje všetky potrebné kroky na tvorbu neurónových sietí. Ako základné jednotky má modely a vrstvy. Vrstvy zapuzdrujú stav a výpočty, môžu sa použiť aj na normalizáciu a vektorizáciu textu. Model zoskupuje vrstvy a následne ho možno trénovať na dátach. Obsahuje tiež rôzne optimalizátory, metriky a stratové funkcie. Keras je navrhnutý s dôrazom na nasledujúce ciele:

- Jednoduché a konzistentné rozrhanie.
- Mimimaliizácia počtu akcií pre bežné použitie.
- Poskytnutie jasných hlásení o chybách.
- Kód je stručný a čitateľný.
- Princíp progresívneho odhaľovania zložitosti - jednoduchý na začatie, zložitejšie postupy sa dajú naučiť za pochodu.

Keras umožňuje jednoduché a rýchle prototypovanie (vďaka používateľským rozhraniam, modulárnosti a možnosti rozšírenia) a podporuje konvolučné siete, rekurentné siete a ich kombinácie [29].

5.3 Gensim

Gensim je taktiež open-source knižnica jazyka Python, ktorá slúži na reprezentáciu dokumentov ako sémentických vektorov. Gensim je navrhnutý na spracovanie nespracovaných, neštruktúrovaných digitálnych textov pomocou algoritmov strojového učenia bez dozoru. Poskytuje prístup k algoritmom ako sú Word2Vec a FastText.

5.4 Hugging Face

Hugging Face je spoločnosť zameraná na umelú inteligenciu, je známa svojimi pokročilými technológiami transfer learningu v oblasti NLP. Ich platforma umožňuje vývojárom využívať natrénované modely a aplikovať ich na širokú škálu úloh bez potreby od základu trénovať nové modely. Modely je možné podľa potreby upravovať, meniť váhy a spraviť finetuning, na prispôsobenie pre danú úlohu.

5.5 Projekt Gutenberg

Projekt Gutenberg je iniciatíva, ktorá sa zameriava na digitalizáciu a poskytovanie voľne dostupných elektronických verzií literárnych diel. Texty v Project Gutenberg sú zväčša knihy, ktorých autorské práva už vypršali, čo znamená, že sú vo verejnej doméne. Tieto texty sú potom digitalizované a poskytované vo formátoch ako sú textové súbory, ePub alebo PDF, aby boli ľahko dostupné pre čítanie a šírenie. Project Gutenberg je významným zdrojom literárnych diel a je často využívaný na výskum, vzdelávanie a rekreačné čítanie. Jeho kolekcia obsahuje vyše 70 000 kníh v rôznych jazykoch [30].

5.6 Ďalšie použité knižnice

Okrem vyššie spomínaných technológií, boli v diplomovej práci využité ďalšie podporné knižnice, najmä na preprocessing dát, exploratívnu analýzu a vizualizácie, napríklad:

- **NumPy** a **Pandas**: na manipulaciu s dátami a ich analýzu.
- **Matplotlib** a **Seaborn**: vizualizácia dát a výsledkov.
- **Scikit-learn**: na úlohy strojového učenia, rozdelenie dát, normalizácia a výpočet metrík.
- **NLTK (Natural Language Toolkit)**: preprocessing textu, ako tokenizácia, odstráňovanie stopwords.
- **RE (Regular Expression)**: odstráňovanie non-ASCII znakov, čísel a interpunkcie.

Kapitola 6

Implementácia

6.1 Datasety

Na sťahovanie dát z projketu Gutenberg 5.5 bol využitý program, ktorý v rámci tvorby svojej diplomovej práce vytvoril Ing Vojtěch Prokop [31]. Išlo o automatizované sťahovanie diel z Projektu Gutenberg. Tento kód bol vytvorený pomocou Pythonu a jazyka R. Kód som mierne modifikovala podľa potrieb. Všetky diela boli v anglickom jazyku.

Na vytvorenie konkrétneho datasetu som do projektu doplnila dva Python scripty. Prvý *prepare_data.py* prešiel všetky stiahnuté diela a vybral z nich x slov, s tým, že vyberal celé odseky a každý odsek musel mať minimálne y slov. Toto obmedzenie bola pridané, aby neboli pridávanie krátke odseky priamej reči, napríklad len *"Yes?"* a podobne. Odseky sú tiež vyberané z diel náhodne, nejde o odseky idúce po sebe, aby dataset obsahoval komplexnejšiu slovnú zásobu celého diela. Počet slov x bol zvolený podľa toho, aký veľký dataset bolo potrebné vytvoriť. Z tohto scriptu vznikol 1 txt súbor pre každé dielo. Druhý script *create_dataset.py* má za úlohu tieto vzniknuté súbory spojiť a vytvoriť dataset vo formáte csv, ktorý má dva stĺpce - author_id a text.

6.2 Rozdelenie dát

Pred trénovaním modelov sa data ešte delia na 3 sady - trénovacia (72%), testovacia (20%) a validačná (8%). Trénovacia sada sa používa k učeniu modelov, modely sa v nej snažia nájsť vzorce. Validačná sada sa používa na overenie výkonnosti modelu počas tréningu. Model sa trénuje na trénovacej sade a súčasne sa po každej epoche vykonáva hodnotenie modelu na validačnej sade. Týmto sa dá odhaliť overfitting, pokiaľ model podáva dobre výsledky na trénovacej sade, ale nevie urobiť dobrú klasifikáciu na validačnej sade, teda na dátach, ktoré predtým nevidel. Testovacia sada sa používa na konečné vyhodnotenie modelu po skončení trénovania.

6.3 Vývojové prostredie

Pri príprave stiahnutých dát bolo využitý Visual Studio Code a jazyk Python. Implementácia a trénovanie niektorých modelov bolo vykonané na cloudovej platforme Google Colab a tiež na jej rozšírenejšej verzii Google Colab Pro, ktorá ponúka okrem GPU T4 aj výkonnejšie procesory V100 a P100. Google Colab bol veľmi výhodný, keďže som mohla využívať priamy prítup na môj Google Disk, kde som mala uložené všetky potrebné dáta. Niektoré modely boli trénované pomocou vzdialeného prítupu na školský server, bežiaceho na operačnom systéme Linux, z dôvodu obmedzených zdrojov (unit) na Google Colab. Z dôvodu využitia rôznych GPU nemôžem vo všeobecnosti porovnať rýchlosť trénovania na všetkých datasetoch. Aby mohol byť porovnaný čas aspoň medzi modelmi, porovnám ho v rámci jedného datasetu.

6.4 Preprocessing

V rámci preprocessingu dát boli použité niektoré metódy z kapitoly 2.2. Pre modely strojového učenia (Logistická regresia, Naive-Bayes) sa preprocessing nachádza priamo v scripte aj s modelmi. Toto umiestnenie bolo z dôvodu, že tento preprocesing sa nevyužíva pre iné modely a preto ho nebolo potrebné vytvárať osobitne a ukladať ako nový dataset. Konkrétne sa aplikovalo tokenizácia, odstránenie non-ASCII znakov, čísel, interpunkcie a stopwords, prevod všetkých znakov na malé a lemmatizácia. Následne bol text spojený do 1 reťazca.

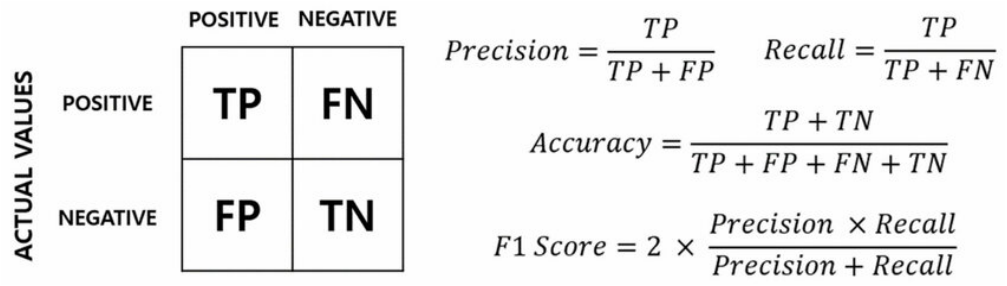
Pre modely neurónových sietí a transformera bol vytvorený na osobitný preprocessing, ktorý sa nachádza v scripte *Preprocessing.py*. Výsledok tohto preprocesingu bol použitý vo všetkých modeloch NN. Dáta boli rozdelené na slová a boli využité metódy odstránenie non-ASCII znakov, čísel a stopwords, prevod na malé písmená a prevod do reťazca. Lemmatizáciia ani stemming nemuseli byť využité, kvôli následnému využitiu word embedding-u, ktorého výsledné vektory sú schopné identifikovať významovú podobnosť slova, aj keď je v rôznych tvaroch.

6.5 Vytvorené modely

Modely boli vytvárané z rôznych odvetví strojového a hlbokého učenia, aby bolo možné ich porovnať a nájsť najlepší model na identifikáciu autora. Predpokladá sa, že Transformer modely by mali fungovať lepšie na zložitejších úlohach a väčších korpusoch, zatiaľ čo jednoduchšie modely by mohli podávať lepšie výsledky (časové aj presnosť) na menších dátach.

Na začiatku testovania bolo vytvorených niekoľko modelov pre každú z kategórií:

- **Strojové učenie**: Logistická regresia, Naive-bayes
- **DNN**: Dense a DropOut vrstvy
- **CNN**: Conv1D vrstva



Obr. 6.1: Confusion matrix [32]

- **RNN**: LSTM, LSTM + GRU, Biderectional LSTM vrstvy
- **Transformer**: Bert, DistilBert, Electra

Boli testované na prvom datasete, ktorý obsahoval 44 autorov a 36-49 diel pre každého z nich. Na tomto datasete boli modely rôzne testované a menené, nakoniec niektoré z nich boli ponechané na ďalšie testovanie. Prebehlo ešte testovanie na druhom datasete a následne boli z modelov vybraté tie, ktoré sa zdali ako najvhodnejšie pre dané data a podávali dotatočne uspokojujúce výsledky. Podrobnejší popis niektorých vybraných modelov bude v časti experimenty.

6.5.1 Vyhodnotenie modelov

Modely boli posudzované na základe confusion matrix a teda 4 metrík:

- **Accuracy**: poskytuje presnosť správne klasifikovaných príkladov ku celkovému počtu príkladov. Táto metrika môže pôsobiť zavádzajúco pri nevyvážených triedach.
- **Precision**: udáva, koľko príkladov z tých, ktoré model klasifikoval ako pozitivné, boli skutočne pozitívne.
- **Recall**: meria, koľko skutočne pozitívnych príkladov bolo správne klasifikovaných.
- **F1 score**: udáva harmonikcký priemer precision a recall. Vyššia hodnota znamená lepšiu rovnováhu medzi nimi, čo môže byť výhodné pri nevyvážených triedach.

Ich vzorce polu s confusion matrix sú zobrazené na obrázku 6.1

Kapitola 7

Experimenty

Experimenty boli prevádzané na 6 rôznych datasetoch, ktoré sa líšia svojou veľkosťou, počtom autorom, konkrétnymi autormi, boli to balancované aj nebalancované datasety. V rámci tejto kapitoly budú predstavené jednotlivé experimenty, ich výsledky a tiež záverečné zhodnotenie.

7.1 Experiment 1

Prvý experiment je zameraný na porovnanie 2 datasetov s rovnakými autormi, ale jeden dataset (*Dataset 1*) bude mať vybalancovanú distribúciu tried (počet diel 58-62) a druhý (*Dataset 2*) bude obshahovať všetky diela autorov (59-215), teda má nevybalancovanú distribúciu tried, ktorá sa častejšie vyskytuje v reálnom svete. Keďže je v rámci tohto experimentu vykonaných dostatok pokusov na všetkých modeloch, budú tu vyhodnotené aj jednotlivé modely spojené s rôznymi vektorizáciami. Datasety sú popísané v tabuľke 7.1. Hodnoty pre počet unikátnych slov a priemerný počet slov sú uvádzané po preprocessingu, teda už vo finálnych dátach, na ktorých boli trénované modely neurónových sietí a transformeru. Modely strojového učenia mali vlastný preprocessing, ako bolo uvedné v kapitole 6.4.

Dataset	Počet autorov	Počet diel (na autora)	Počet unikátnych slov	Priemerný počet slov
Dataset 1	20	58–62	119 219	6160
Dataset 2	20	59–215	147 645	6130

Tabuľka 7.1: Štruktúra balancovaného a nebalanovaného datasetu

V rámci tohto experimentu je možné na základe teoretických poznatkov očakávať, že modely budú na vybalancovaných dátach dosahovať vyššie hodnoty metriky *accuracy*, kvôli rovnomernej reprezentácii všetkých tried. Naopak, pri nebalancovaných dátach by mohlo byť vyššie hodnotenie modelov pomocou *f1 score*, ktorý lepšie zohľadňuje nerovnováhy v počte príkladov pre jednotlivé triedy.

Ďalšou hypotézou je, že pokročilé modely, ako sú neurónové siete a transformery, poskytnú lepšie výsledky pri nebalancovaných dátach. Zvládajú efektívnejšie extrahovať komplexné vzory a mali by zvládať rôznorodosť v trénovacích dátach. Modely strojového učenia, by mohli byť výkonnejšie na vybalancovaných dátach.

7.1.1 Modely strojového učenia

Modely strojového učenia (Logistická Regresia a Naive-Bayes) boli trénované so siedmymi rôznymi vektorizačnými metódami:

- **BOW**: vytvorená pomocou funkcie *CountVectorizer*.
- **TF - IDF**: vytvorená pomococu funkcie *TfidfVectorizer*.
- **GloVe**: bola použitá vektórova reprezentácia natrénovaná na dátovom súbore so 6 miliardami slov a každé slovo je reprezentované vektormi o veľkosti 300 dimenzií.
- **Word2Vec**: bol využitý model predtrénovaný na dátach z Google News a každé slovo je reprezentované vektormi o veľkosti 300 dimenzií.
- **Bert**: bol využitý menší zo základných Bert modelov bert-base-uncesaed.
- **DistilBert**: rovnako bol využitý menší model distilbert-base-uncased.
- **Electra**: podobne bol využitý jeden z menších modelov electra-small-discriminator.

7.1.1.1 Výsledky

Výsledky pre balancované dáta sú zobrazené v tabuľke 7.2 a pre nebalancované v tabuľke 7.3. Výsledky budú okomentované na základe metrík acuracy a f1 score, metriky precision a recall slúžia ako doplňujúce údaje. Ale najlepší výsledok recall, ktorý ukazuje aký podiel skutočných dokumentov daného autora bol správne identifikovaný modelom ako patriaci tomu autorovi, bude zvýraznený v tabuľkách.

Vhodnejší model pre oba datasety bola Logistická regresia. Naive-Bayes poskytol lepšie výsledky iba pri vektorizácia word2vec, pri oboch datasetoch o 9-12%. Naive-bayes si pomerne dobre viedol v spojení s BOW a TD-IDF na balancovaných dátach. Pri nebalancovaných dátach mal problém s presnosťou v spojení s vektorizáciou TF-IDF. Tento experiment bol opakovaný viackrát, aby sa vyvrátilo riziko chyby, ale výsledná presnosť bola vždy 35-41%. To je približne polovica z presnosti, ktorá bola dosiahnutá na balancovanom datasete. Ďalšie väčšie rozdiely, kde bolo dosiahnuté vyššie skóre pri balancovaných dátach, mal Naive-Bayes s word2vec (o cca 14-16%) a s GloVe (o cca 7-8%).

Z vektorizácií bola rozhodne najvhodnejšia BOW. Najlepší výsledok dosiahla Logistická regresia spojená s BOW. Pre balancovaný dataset bol tento výsledok totožný u accuracy aj f1 score **97,1%**.

Pri nebalancovanom datasete to bolo **96,7%**, resp. **96,8%**. Pri Naive-Bayes išlo o niečo menšie výsledky, v rozmedzí 84,6 - 90,8%.

DistilBert poskytoval o niečo lepšie výsledky pri nebalancovaných dátach, ale išlo o maximálne 1,5%. Electra poskytla pri nebalancovyných dátach a modeli Logistickej regresie lepšie výsledky o 3,7 - 4,6%.

Na základe týchto výsledkov je možné zhodnotiť, že balancované dáta sú vo všeobecnosti lepšie, modely tak nemôžu uprednostňovať triedy s väčším počtom záznamov. Pre podobne veľké datasety na úlohu identifikácie autora je vhodné zvoliť vektorizáciu BOW alebo TF-IDF s logistickou regresiou, či už pôjde o balancované alebo nebalancované dáta. Ostatné (embedding) vektorizácie neposkytli v spojení s modelmi strojového učenia veľmi dobre výsledky.

Vektorizácia	**Model**	**Accuracy (%)**	**F1 Score (%)**	**Precision (%)**	**Recall (%)**
BagOfWords	Log. Regresia	97,1	97,1	97,4	**97,1**
TF-IDF	Log. Regresia	96,7	96,6	97,0	96,7
GloVe	Log. Regresia	82,1	82,3	84,2	82,1
Word2Vec	Log. Regresia	67,1	67,1	81,8	67,1
Bert	Log. Regresia	80,8	81,1	83,2	80,8
Distilbert	Log. Regresia	76,3	76,2	77,5	76,3
Electra	Log. Regresia	64,2	64,7	67,2	64,2
BagOfWords	Naive-Bayes	90,8	88,8	88,3	90,8
TF-IDF	Naive-Bayes	79,6	77,6	83,0	79,6
GloVe	Naive-Bayes	78,3	78,8	80,5	78,3
Word2Vec	Naive-Bayes	79,2	79,2	80,3	79,2
Bert	Naive-Bayes	70,0	70,2	72,6	70,0
Distilbert	Naive-Bayes	59,6	59,6	64,9	59,6
Electra	Naive-Bayes	43,8	42,5	44,9	43,8

Tabuľka 7.2: Výsledky modelov strojového učenia na balancovaných dátach.

7.1.2 Modely neurónových sietí

Pri modeloch neurónových sietí sa testovali 3 rôzne architektúry - DNN, CNN a RNN. Model RNN, ktorý obsahoval len vrstvu LSTM nebol ďalej testovaný pre embedding vektorizácie, pretože poskytoval zo všetkých RNN sietí najhoršie výsledky. Všetky ostatné modely boli trénované s TF-IDF, FastText a Glove. BOW vektorizácia bola vyskúšana len pri DNN, pretože ostatné modely na ňu neboli prispôsobené, ale kvôli skvelým výsledkom pri predchádzajúcich modeloch som ju chcela zahrnúť.

7.1.2.1 Výsledky

Výsledky pre balancované dáta sú zobrazené v tabuľke 7.4 a pre nebalancované dáta v tabuľke 7.5. Keďže pôvodný predpoklad, že f1 score a accuracy budú rozdielne v predchádzajúcich modeloch

Vektorizácia	Model	Accuracy (%)	F1 Score (%)	Precision (%)	Recall (%)
BagOfWords	Log. Regresia	96,7	96,8	97,3	**96,7**
TF-IDF	Log. Regresia	94,6	94,4	95,3	94,6
GloVe	Log. Regresia	80,1	78,9	83,5	80,1
Word2Vec	Log. Regresia	56,3	51,1	56,4	56,3
Bert	Log. Regresia	78,0	77,4	78,0	78,0
Distilbert	Log. Regresia	76,7	76,3	78,1	76,7
Electra	Log. Regresia	68,8	68,4	70,0	68,8
BagOfWords	Naive-Bayes	86,4	84,6	90,6	86,4
TF-IDF	Naive-Bayes	40,4	35,3	50,1	40,4
GloVe	Naive-Bayes	71,1	70,6	79,0	71,1
Word2Vec	Naive-Bayes	65,2	63,1	73,0	65,2
Bert	Naive-Bayes	65,0	65,0	68,4	65,0
Distilbert	Naive-Bayes	61,1	59,9	63,4	61,1
Electra	Naive-Bayes	38,4	34,6	43,0	38,4

Tabuľka 7.3: Výsledky modelov strojového učenia na nebalancovaných dátach.

strojového učenia a rovnako ani v modeloch neurónových sietí neplatí, budem ďalej vyhodnocovať modely len na základe metriky accuracy.

Ako najpresnejšia architektúra vyšla DNN pre oba datasety. Podávala veľmi dobré výsledky s vektorizáciami BOW , TF-IDF aj s GloVe, pri FastText už boli výsledky nižšie. Pri CNN podobne boli najlepšie výsledky dosiahnuté s TF-IDF, následoval GloVe a potom FastText.

Pre RNN modely s bidirectional LSTM vrstvou dosiahol najlepšie výsledky GloVe, následne TF-IDF a posledný bol FastText. Pri modeloch s vrstvami LSTM aj GRU boli všeobecne dosahované veľmi nízke výsledky. Jedine na nebalancovaných dátach sa podarilo dosiahnuť v spojení s FastText accuracy 65,7%, čo bolo 3-5x väčšie skóre ako pri ostatných pokusoch s týmito vrstvami.

Pri porovnaní balancovaných a nebalancovaných dát, sú výsledky opačné ako pri modeloch strojového učenia. Všeobecne lepšie výsledky ukázali nebalancované dáta. Ukazuje to, že tieto modely sú už komplexnejšie a zvládajú lepšie dáta z reálneho sveta. Ďalším dôvodom môže byť, že pre nebalancované dáta bol celkový počet diel väčší (pre väčšinu autorov), takže sa modely mohli lepšie natrénovať.

Najlepší dosiahnutý výsledok bol pri oboch datasetoch dosiahnutý s modelom DNN a vektorizáciou BOW. Na balancovaných dátach to bolo **95,0%** a na nebalancovaných **97,2%**.

7.1.3 Transformer modely

Pri modeloch Transformer boli na počiatočných datasetoch testované rôzne hodnoty pre learning rate (kapitola 7.2) a ako najlepšie vyšli hodnoty *5e-5* a *8e-5*. S týmito hodnotami boli potom testované všetky 3 typy architektúry. Modely boli vybraté malé - rovnaké ako pri strojovom učení popísané v kapitole 7.1.1. Opäť sú pre oba datasety priložené tabuľky s výsledkami, pre balanco-

Model	Vektorizácia	Accuracy (%)	F1 Score (%)	Precision (%)	Recall (%)
DNN	BagOfWords	95,0	94,4	95,3	**94,4**
DNN	TF-IDF	93,3	93,4	94,5	93,3
DNN	FastText	68,8	67,4	76,6	68,8
DNN	GloVe	86,2	86,2	87,3	86,3
CNN	TF-IDF	92,9	93,0	94,3	92,9
CNN	FastText	80,0	79,2	80,2	80,0
CNN	GloVe	86,7	86,7	88,7	86,7
LSTM	TF-IDF	10,8	9,4	13,2	10,8
Bidirectional LSTM	TF-IDF	77,9	77,6	81,0	77,9
Bidirectional LSTM	FastText	49,6	46,0	49,4	49,6
Bidirectional LSTM	GloVe	82,1	81,9	83,3	82,1
LSTM+GRU	TF-IDF	12,1	8,3	11,9	12,1
LSTM+GRU	FastText	16,3	12,2	13,1	16,3
LSTM+GRU	GloVe	13,3	11,9	18,3	13,3

Tabuľka 7.4: Výsledky modelov neurónových sietí pre balancované dáta

Model	Vektorizácia	Accuracy (%)	F1 Score (%)	Precision (%)	Recall (%)
DNN	BagOfWords	97,2	96,3	96,7	96,2
DNN	TF-IDF	96,4	96,4	96,7	**96,4**
DNN	FastText	67,1	65,5	70,2	67,1
DNN	GloVe	88,5	88,4	89,7	88,5
CNN	TF-IDF	93,6	93,7	94,6	93,6
CNN	FastText	80,4	80,4	82,1	80,4
CNN	GloVe	90,0	90,2	91,3	90,0
LSTM	TF-IDF	17,9	14,0	19,2	17,9
Bidirectional LSTM	TF-IDF	78,3	78,5	82,9	78,3
Bidirectional LSTM	FastText	63,3	61,2	64,1	63,3
Bidirectional LSTM	GloVe	88,0	88,0	89,0	88,0
LSTM+GRU	TF-IDF	65,7	65,5	75,5	65,7
LSTM+GRU	FastText	13,3	10,4	13,1	13,3
LSTM+GRU	GloVe	23,3	20,4	26,3	23,3

Tabuľka 7.5: Výsledky modelov neurónových sietí pre nebalancované dáta

vané dáta tabuľka 7.6 a pre nebalancované dáta tabuľka 7.7. Vyhodnocovanie opäť prebehne podľa metriky accuracy.

Pri výbere najlepšej architektúry a najlepšieho modelu to v tomto prípade nie je úplne jednoznačné medzi modelom Bert a DistilBert. Na balancovaných dátach podáva o 2-4% lepšie výsledky Bert, pri nebalancovaných je to o tesných 0,5-1% DistilBert. Architektúra Electra mala pre oba datasety najhoršie výsledky. Ideálna hodnota learning rate sa taktiež nedá určiť všeobecne. Pre modely Bert a DistilBert je lepšia hodnota 5e-5, pre model Electra to je 8e-5,

Každopádne vo všetkých prípadoch sú dosiahnuté lepšie výsledky pre nebalancované dáta, čo

môže byť opäť spojené s tým, že dáta pre jednotlivých autorov boli rozsiahlejšie a modely sa vedeli na nich lepšie učiť.

Najlepší výsledok pre balancované dáta je model Bert s learning rate 5e-5 a s accuracy **80,0%**. Pre nebalancované dáta je to model DistilBert a learning rate 5e-5 a accuracy **86,7%**.

Model	**Learning Rate**	**Accuracy (%)**	**F1 Score (%)**	**Precision (%)**	**Recall (%)**
BERT	5e-5	80,0	79,6	81,1	**79,4**
BERT	8e-5	77,5	76,8	80,2	76,3
DistilBERT	5e-5	77,9	76,4	79,4	76,4
DistilBERT	8e-5	73,8	73,7	78,0	73,7
Electra	5e-5	58,3	55,0	56,0	57,9
Electra	8e-5	63,8	60,5	66,3	62,9

Tabuľka 7.6: Výsledky transformer modelov pre balancované dáta

Model	**Learning Rate**	**Accuracy (%)**	**F1 Score (%)**	**Precision (%)**	**Recall (%)**
BERT	5e-5	85,7	84,3	84,6	85,2
BERT	8e-5	84,9	84,0	86,2	85,4
DistilBERT	5e-5	86,7	86,1	86,2	**86,7**
DistilBERT	8e-5	85,4	83,6	84,1	85,2
Electra	5e-5	67,3	60,2	65,5	61,7
Electra	8e-5	68,8	60,4	64,0	62,1

Tabuľka 7.7: Výsledky transformer modelov pre nebalancované dáta

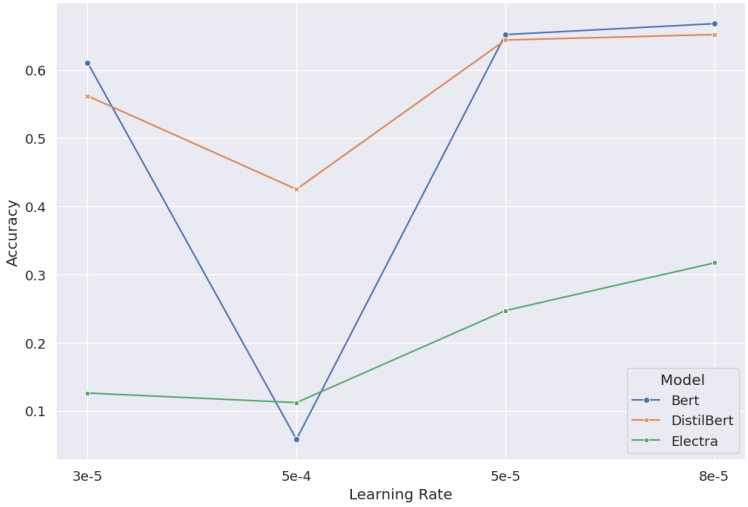
7.1.4 Zhodnotenie

V rámci predpokladov bolo zmienené, že f1 score by mohlo lepšie fungovať na nebalancovaných dátach a accuuracy na balancovaných. Tento predpoklad bol chybný. Skóre týchto metrík bolo vo väčšine prípadov veľmi podobné a nedominovala ani jedna metrika pri žiadnych modeloch.

Druhý predpoklad bol potvrdený. Modely strojového učenia boli presnejšie na balancovaných dátach, modely NN a transformer na nebalancovaných dátach. Najpresnejšie modely pomedzi všetky architektúry sú zobrazené v tabuľke 7.8. Celkové zhodnotenie experimentov bude uvedené v závere tejto práce.

Dataset	**Model**	**Vektorizácia**	**Accuracy (%)**
Nebalancovaný	DNN	BOW	97,2
Balancovaný	Logistická Regresia	BOW	97,1
Balancovaný	Logistická Regresia	TF-IDF	96,7
Nebalancovaný	Logistická Regresia	BOW	96,7

Tabuľka 7.8: Najlepšie modely v experimente 1



Obr. 7.1: Výsledky transformer modelov na základe rôznych learning rate

7.2 Experiment 2

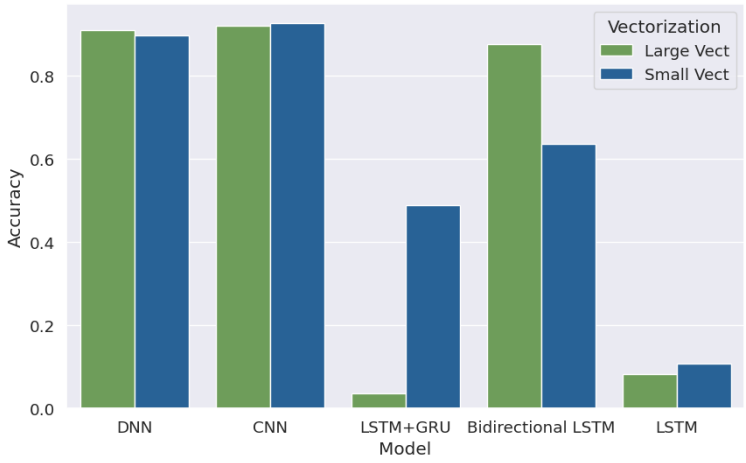
V tomto experimente bol cieľ získať čo najlepšiu hodnotu learning rate. Learning rate je jeden z hyperparametrov, určuje rýchlosť, akou sa parametre modelu upravujú alebo aktualizujú. Príliš nízka hodnota môže viesť k pomalému učeniu, príliš vysoká hodnota môže spôsobiť nestabilné učenie. Hodnoty learning rate boli nastavené na *3e-5*, *5e-4*, *5e-5* a *8e-5* a boli testované na všetkých troch Transformer modeloch. Výsledky sú zobrazené na obrázku 7.1

Na výsledkoch je vidieť, že správne nastavenie learning rate je veľmi dôležité, pretože môže výrazne ovplyvniť presnosť modelov. Najhoršie výsledky pre každý model dosiahla najväčšia hodnota 5e-4. Pri Bert modeli bola accuracy dosiahnutá s touto learning rate len 6%, pričom najlepšia accuracy pre Bert bola takmer 67%. Na základe výsledkov sa najviac osvedčila hodnota **8e-5**, pre každý model a za ňou nasledovala hodnota **5e-5**. Tieto 2 hodnoty sa teda budú používať na trénovanie modelov na iných datasetoch. Podrobné výsledky pre všetky modely sú zobrazené v tabuľke A.1 (príloha A).

7.3 Experiment 3

Tento experiment sa zameriava na testovanie nastavení TextVectorization vrtsvy. Testovali sa modely neurónových sietí (DNN, CNN, RNN) na 2 rôznych nastaveniach nasledujúcich parametrov:

- **Max_tokens**: predstavuje počet najčastejších unikátnych slov, ktoré sa budú brať do úvahy. Väčší slovník môže zobrať do úvahy viac slov, ale niektoré z nich môžu byť nevýznamné.
- **Output_sequence_length**: určuje dĺžku vektorovej reprezentácie pre každý vstup. Ak bude vstup dlhší, tak sa oreže, pokiaľ bude kratší doplní sa pomocou padding-u. Dĺžka by mala byť



Obr. 7.2: Výsledky modelov s rôznym nastavením vektorizácie

dosť dlhá na to, aby zachytila kontext, ale nie príliš dlhá, aby sa zbytočne nespracovávali doplnené nulové hodnoty.

- **Embedding_dim**: predstavuje veľkosť priestoru vektorovej reprezentácie, v ktorom budú reprezentované tokeny. Vyšší rozmer poskytne priestor zachytiť viac informácií pre každé slovo, ale zvyšuje výpočtovú náročnosť.

Dataset je zložený z 20 autorov, každý má 58-62 diel, skladá sa z 143 223 unikátnych slov a priemerná dĺžka pre jedno dielo je 9 490 slov.

Konkrétne nastavenia sú zobrazené v tabuľke 7.9. Hodnoty boli vybrané vzhľadom na počet unikátnych slov v datasete a na priemernú dĺžku jedného diela.

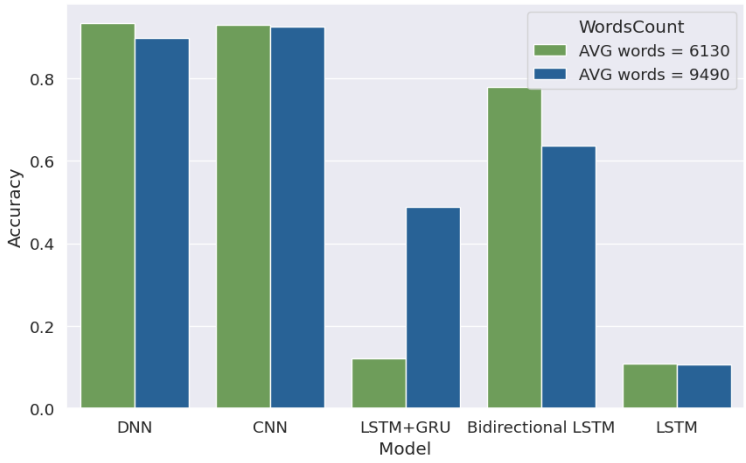
Max_tokens	Output_sequence_length	Embedding_dim
60 000	6000	256
90 000	9000	512

Tabuľka 7.9: Nastavenie parametrov pre TextVectorization

7.3.1 Výsledky

Výsledky pre jednotlivé modely sú zobrazené na obrázku 7.2, podrobnejšie výsledky sú ukázané aj v tabuľkách A.3 pre menšiu vektorizáciu a A.2 pre väčšiu.

Modely CNN a DNN mali pre obe vektorizácie podobné výsledky a s presnosťou okolo 90% poskytli uspokojivé výsledky. Model RNN s LSTM vrstvou ponúkol opäť veľmi nízke výsledky a o niečo lepšie si viedol s menšou vektorizáciou, ale išlo o minimálny rozdiel. Ďalšie 2 modely RNN ukázali presne opačné výsledky. Model s LSTM a GRU vrtsvou si viedol výrazne lepšie s menšou vektorizáciou, kde dosiahol presnosť takmer 50%. S väčšou vektorizáciou dosahoval v podstate nulovú



Obr. 7.3: Výsledky modelov s rôznym počtom slov pre diela

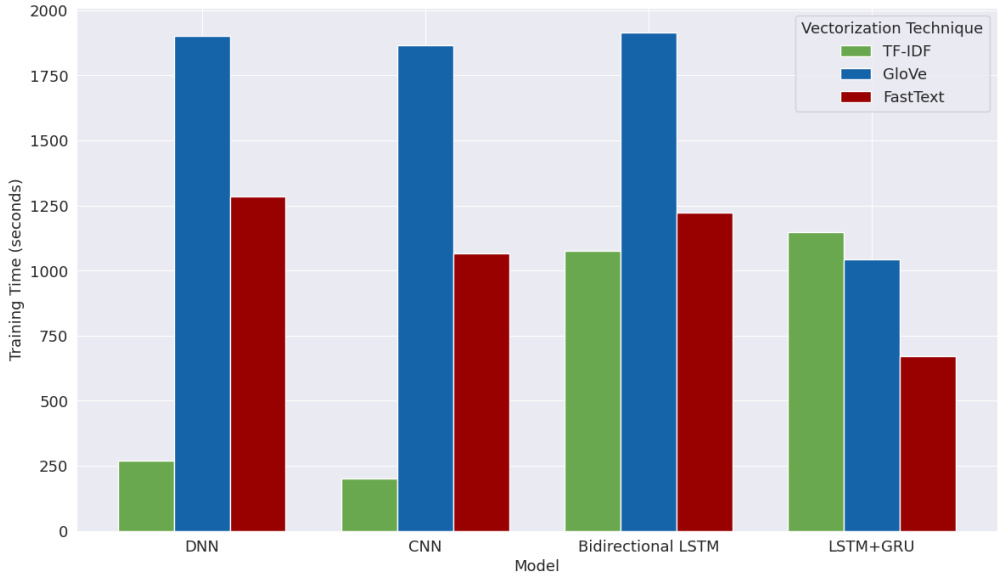
prenosť. Tieto modely boli spúšťane opakovane a výsledky boli stále rovnaké. Model s Bidirectional LSTM vrstvou dosahoval o cca 24% lepšie výsledky pri väčšej vektorizácií a priblížil sa k výsledkom DNN a CNN modelov.

Podľa výsledkov je vidieť, že jednoduchá odpoveď na otázku, či je lepšia väčšia alebo menšia vektorizácia neexistuje, ani vzhľadom na konkrétne dáta. Počas vykonávania ostatných experimentov sa mi potvrdilo, že nastavovať dĺžku vektorovej reprezentácie pre každý vstup podľa priemerného počtu slov (v každom vstupe) je výhodné a väčšinou to bolo ideálne nastavnie tohto parametra. Čo sa týka nastaveniu počtu unikátnych slov, tam výsledky neboli jednoznačné. Okrem prispôsobenia pre dáta je potrebné vektorizáciu prispôsobiť aj konkrétnym modelom.

7.4 Experiment 4

Tento experiment je zameraný na počet slov v jednotlivých dielach. Balancovaný dataset v experimente 1 a dataset z experiemntu 3 sa skladajú z rovnakých autorov a majú rovnaký počet diel pre každého autora, líšia sa len počtom slov pre jednotlivé diela. Preto na základe ich výsledkov je možné porovnať, či počet slov pre každé dielo nejako ovplyvní presnosť. Trénované sú na menšej vektorizácií z experiemntu 3. Priemerný počet slov po preprocessingu je v datasetoch 6130 a 9490. Výsledky sú zobrazené na obrázku 7.3.

Modely DNN a CNN opäť ukazujú, že podávajú konzistentné výsledky. Model s LSTM vrstou neponúkol žiadne uspokojivé výsledky. Pri ďalších 2 modeloch RNN to bolo opačne, podobne ako v predchádzajúcom experimente. Bidirectional LSTM model si viedol lepšie pri menšom priemernom počte slov a LSTM + GRU model pri väčšom priemernom počte slov.



Obr. 7.4: Porovnanie času potrebného na natrénovanie modelov NN

7.5 Experiment 5

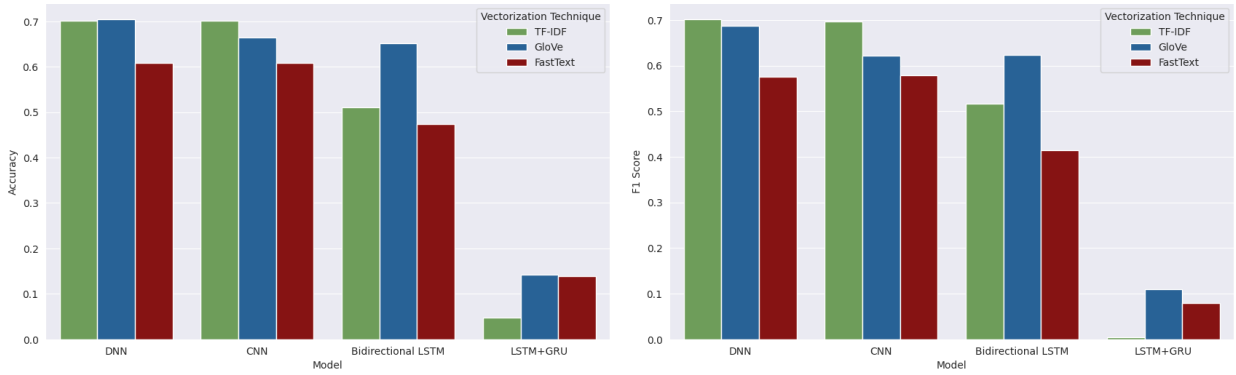
V tomto experimente a bude primárne hodnotiť čas potrebný na natrénovanie modelov neurónových sietí a strojového učenia s rôznymi vektorizáciami.

7.5.1 Modely neurónových sietí

Všetky experimenty boli trénované na 15 epochách. 1 z modelov RNN sa zastavil po 13 epochách kvôli overfitting-u, jeho čas bol na účely tohto experimentu prepočítaný na 15 epoch pomocou priemernej dĺžky jednej epochy. Dataset bol nebalancovaný a skladal a z 81 autorov s rôznym počtom diel (5-160). Výsledné časy sú zobrazené na obrázku 7.4.

Podrobné výsledky pre všetky modely a všetky metriky sú uvedené v tabuľke A.4. Pri 3 modeloch (DNN, CNN, Bidirectional LSTM) trval tréning najkratší čas v spojení s vektorizáciou TF-IDF. Pri RNN modeloch bol tento čas vyšší, oproti jednoduchším modelom. To môže byť spôsobené rekurentnou architektúrou, ktorá má v sebe cykly a pamätá si informácie z predchádzajúcich vrstiev (kapitola 3.2.3).

Pri vektorizácií pomocou GloVe trval tréning najdlhší čas, pri FastText boli časy vyššie v porovnaní s TF-IDF (okrem modelu LSTM+GRU). Práca s word embddingmi je teda časovo náročnejšia a modely potrebujú dlhší čas pre natrénovanie. Aby bolo možné porovnať pomer presnosť - čas, na obrázku 7.5 sú zobrazené hodnoty accuracy a f1 score pre všetky modely. Pri porovnaní presnosti pre word embedding je vidieť, že GloVe je pre tieto dáta lepšia voľba ako FastText.



(a) (b)

Obr. 7.5: Výsledky metrík Accuracy (a) a F1 Score (b) pre rôzne vektorizácie

Z týchto výsledkov vychádza, že pre modely DNN a CNN je najlepšie voľba TF-IDF. Trénovanie s touto vektorizáciou trvá výrazne kratší čas a výsledky sú s ňou najlepšie. Pre RNN modely je lepšia voľba GloVe, kde čas trénovania je síce vyšší, ale presnosť tiež. Model LSTM+GRU dosahuje opäť veľmi nízke výsledky a ani pri jeho úpravach a rôznych nastaveniach hyperparametrov vo vrstvách sa ho nepodarilo vylepšiť. Aspoň nejaké zlepšenie nastalo pri použití word embeddingov GloVe a FastText.

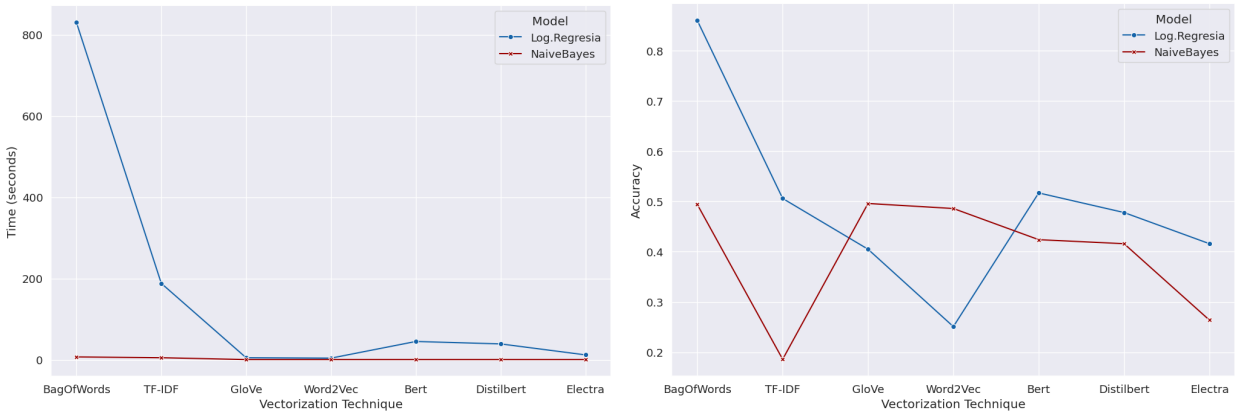
7.5.2 Modely strojového učenia

Modely strojového učenia boli tiež natrénované na tomto datasete. Accuracy aj časová náročnosť modelov je odprezentovaná na obrázku 7.6 a podrobné výsledky sú v tabuľke A.5.

Výsledky ukazujú, že tieto modely nie sú vhodné pre takéto zložitejšie a veľmi nebalancované datasety. To odkazuje aj na experiment v kapitole 7.1, kde síce výsledky neboli také nízke, ale v porovnaní s balancovanými dátami boli horšie. Predpokladám, že vtedy dosiahli modely ešte pomerne dobre výsledky, pretože od každého autora bol minimálny počet diel 58. Takže aj keď dáta neboli balancované, mali modely aspoň dosť dát na naučenie sa vzorov z textu. V tomto datasate bol minimalný počet diel 5, takže tu bol výrazne menší priestor na naučenie sa vzorcov. Tiež mohlo dôjsť k uprednostňovaniu autorov s veľkým počtom diel.

Časovo menej náročný je model Naive-Bayes, ktorého čas potrebný na natrénovanie tohto datasetu bol len v jednotkách sekúnd pri všetkých vektorizačných technikách. Jeeho presnosť však bola nízka, napríklad pri vektorizácií TF-IDF doiahol prenosť menej ako 20%. Najlepšie fungoval s BoW a s word embedding technika GloVe a word2vec, kde dosahoval takmer 50%. Naive-Bayes teda nie je pre takýto typ dát vhodný, napriek svojej rýchlosti.

Časová náročnosť pre Logistickú regresiu je o niečo väčšia, ale len pre niektoré vektorizácie. Najviac časovo náročný bol tréning vektorizáciou BoW, ale zároveň poskytol aj najlepšie výsledky



(a) (b)

Obr. 7.6: Čas trénovania (a) a accuracy (b) pre modely strojového učenia

v tomto pokuse, **86%**. Ostatné výsledky neboli veľmi presné. Transformer architektúry dosiahli priemerné výsledky. Ako najlepší sa javí Bert, najslabšie výsledky dosiahla Electra.

Podľa týchto výsledkov je pre dáta podobného typu ideálne zvoliť Logistickú regresiu a BoW vektorizáciu. To je podobný výsledok, aký bol dosiahnutý aj v experimente v kapitole 7.1.

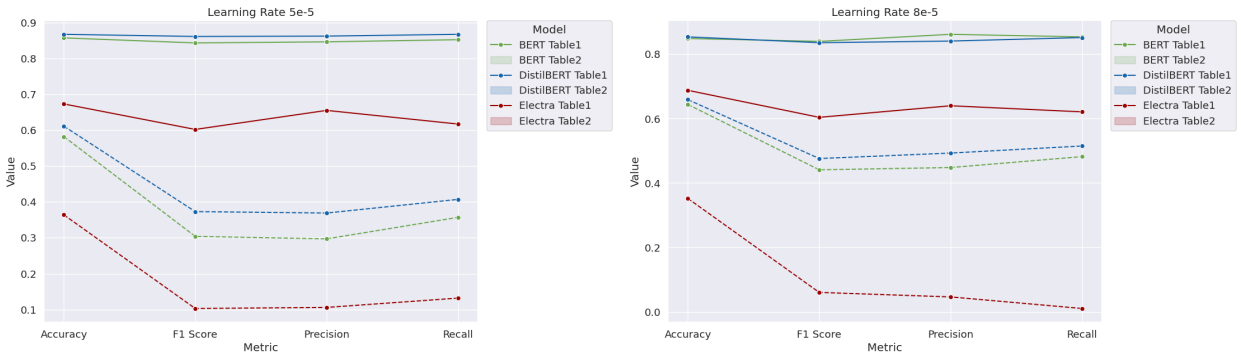
7.6 Experiment 6

Tento experiment využije výsledky z predchádzajúcicch experimentov a otestujem odolnosť Transformer modelov. V prvom experimente 7.1 bolo ukázané, že všetky 3 testované Tranformer modely si viedli lepšie na nebalancovaných dátach. V experimente 7.5 bol použitý tiež nebalancovaný dataset, ale s menej dielami a modely strojvého učenia aj modely neurónových sietí dosahovali s ním nižšiu prenosť ako v predchádzajúcich experimentoch. Preto tento dataset bol otestovaný aj na modeloch Bert, DistilBert a Electra. Jeho výsledky sú zobrazené na obrázku 7.7 a v tabuľke 7.10, ktorá je na grafe označená ako Table2, a ako table1 je myslená tabuľka 7.7 a jej výsledky.

Model	**Learning Rate**	**Accuracy (%)**	**F1 Score (%)**	**Precision (%)**	**Recall (%)**
BERT	5e-5	58,2	30,4	29,7	35,7
BERT	8e-5	64,4	44,1	44,8	48,2
DistilBERT	5e-5	61,1	37,3	36,9	40,7
DistilBERT	8e-5	65,9	47,6	49,3	51,5
Electra	5e-5	36,4	10,3	10,6	13,2
Electra	8e-5	35,2	6,0	4,6	1,0

Tabuľka 7.10: Výsledky transformer modelov

Na priložených grafoch je vidieť, že aj tieto modely sú veľmi náchylné na dataset. Priebeh bol pri oboch datasetoch podobný, v zmysle poradia podľa presností. Model Electra bol vždy najme-



(a) (b)

Obr. 7.7: Výsledky Transformer modelov pri learning rate 5e-5 (a) a 8e-5 (b)

nej presný. Vo väčšine prípadov mal najlepšie výsledky DistilBert, ktorý je výhodný aj kvôli jeho menšej veľkosti a teda menšej časovej náročnosti. Podľa výsledkov je tiež jasné, že ani tieto modely nie sú odolné voči nevhodnému datasetu. V tomto prípade zrejme hralo rolu nedostatok dát pre jednotlivých autorov.

7.7 Experiment 7

V tomto exprimente bol vytvorený dataset podľa konkrétnych autorov. 10 autorov písalo v období medzi rokmi 1550-1700 a ďalších 10 písalo v rokoch 1850-1980. Máme teda 2 skupiny autorov:

- **1550-1700**: *17, 65, 204, 410, 505, 791, 1181, 1230, 1758, 5166*

  Patria tu autori ako William Shakespeare, Samueal Pepys alebo Miguel de Cervantes Savedra.

- **1850-1980**: *45, 69, 120, 213, 585, 594, 1060, 1626, 3141, 8137*

  Patria tu autori ako Winston Churchill, Jack London alebo William Le Queux.

Knihy boli zahrnuté všetky, takže v rozmedzí 9 - 86. Cieľom je preskúmať, či sa budú chybne viac zamienať autori v rámci svojho obdobia alebo na tom nebude záležať. Štýl písania sa mohol za tých 150-300 rokov výrazne zmeniť, čo by mohlo zabrániť chybnému označovaniu autorov pomedzi obdobia. Po každom tréningu modelu sa vykreslila confusion matrix, resp. heatmapa. Z výsledkov vyberiem všetky dvojice, ktoré boli zamienané v rámci jedného behu modelu 4 alebo viackrát. Dvojica bude myslená ako skutočný autor - predpovedaný autor.

Do úvahy sa brali modely DNN, CNN a Bidirectional LSTM s vektorizáciami TF-IDF, GloVe a FastText, dokopy teda 9 modelov. Výsledky tréningu sú v tabuľke A.6. V tabuľke 7.11 sú zobrazené nesprávne predikcie autorov. Ani jedna z chybných predikcií nemieša autorov z 2 obodbí. Zaujímave je aj, že iba 1 chybne predikovaná dvojica je zo skoršieho obdobia (*791, 505*), všetky ostatné sú z neskoršieho obdobia. Autor 505 je Miguel de Cervantes Saavedra a jeho najznámejšie dielo je

Dômyselný rytier Don Quijote de la Manch. Druhým autorom je Molière a jeho naznámejšie diela sú Tartuffe, Misantrop, Lakomec a Zdravý nemocný. Diela od oboch autorov často kritizujú ľudskú povahu a zaužívané spoločenské normy pomoocu komédie, irónie a satiry. Tiež ich diela museli byť do angličtiny prekladané. Tieto faktory mohli viesť k častým zámenám týchto autorov. Pri modernejších autorov mali najviac zámien Carolyn Wells a Edward Stratemeyer. Obaja boli americkí spisovatelia, takže mohli používať podobní slovník. Obaja písali skôr pre mladých čitateľov, a ich knihy boli zamerané na detské a mládežnícke publikum. Na základe týchto podobností si ich modely mohli jednoduchšie zamieňať.

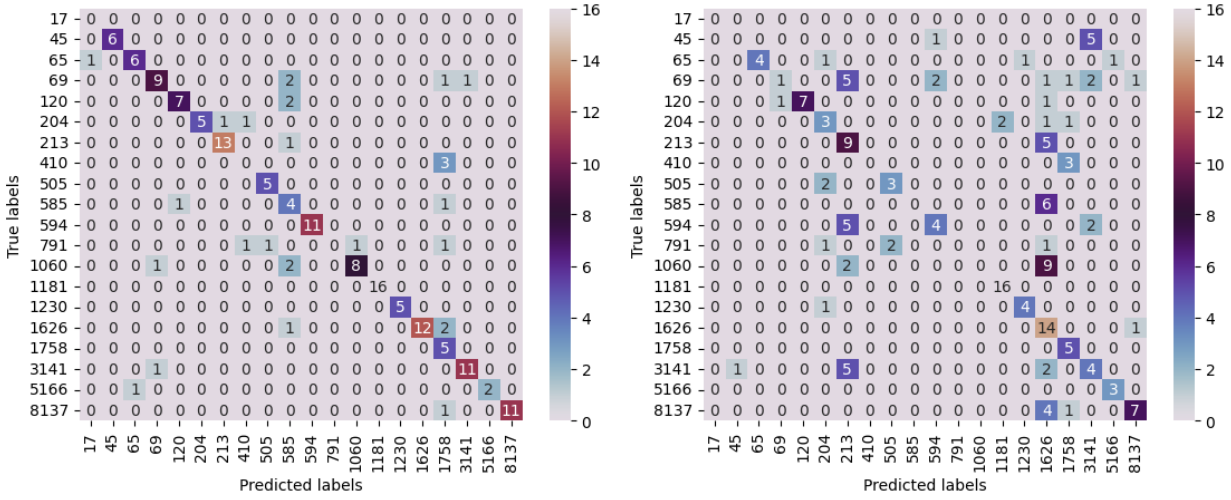
Expriment ukázal, že štýl písania sa medzi obdobiami veľmi zmenil. Modely dokázali správne rozlišovať medzi obdobiami, ale autori z rovnakého obdobia a s podobným žánrom im robili problém. To by možno mohli vyriešiť rozsiahlejšiee dátá, prípadne viac diel. Ja som použila všetky diela, ktoré som mala dostupné z projketu Gutenberg a z každého diela som vzala približne 100 000 slov. Po preprocessingu bola priemerný počet slov na 1 dielo 9230. Ukážka confusion matrix najpresnejšieho modelu, teda CNN s TF-IDF vektorizáciou, a najmenej presného modelu, Bidirectional LSTM s FastText je na obrázku 7.8

Skutočný autor	**Predpovedaný autor**	**Počet zámien**
791	505	(4 + 4 + 4) = 12
1060	1626	(5 + 9) = 14
1060	213	4
69	594	5
69	213	5
45	3141	5
213	1626	5
594	213	6
585	1626	6
8137	1626	4
3141	213	3

Tabuľka 7.11: Chybne predpovedaní autori

7.7.1 Model LSTM+GRU

Vo vyhodnocovaní tohto experimentu nebol zahrnutý model LSTM+GRU, ktorý je zvyčajne súčaťou testovacích modelov, aj napriek svojím nízkym presnotiam. Tentokrát bol testovaný, ale nebol zaradený do skúmaných výsledkov, pretože opäť podal veľmi neupokojivé výsledky a bolo by to pre experiment skresľujúce. Avšak, objavila sa pri ňom zaujímavá vec, ktorá je zobrazená na obrázku 7.9. V oboch prípadoch model vybral rovnakého autora, ktorého skoro stále predikoval (autor 1626). Tento autor mal druhý najvyšší počet diel a to 74. Vzhľadom nato, že ostatné modely fungujú s týmito modelmi relatívne dobre, dataset ani jeho spracovanie by nemalo byť problém. Model sa



(a) (b)

Obr. 7.8: Confusion matrix najpresnejšieho modelu CNN s TF-IDF (a) a najmenej presného modelu Biderectional LSTM s FastText (b)

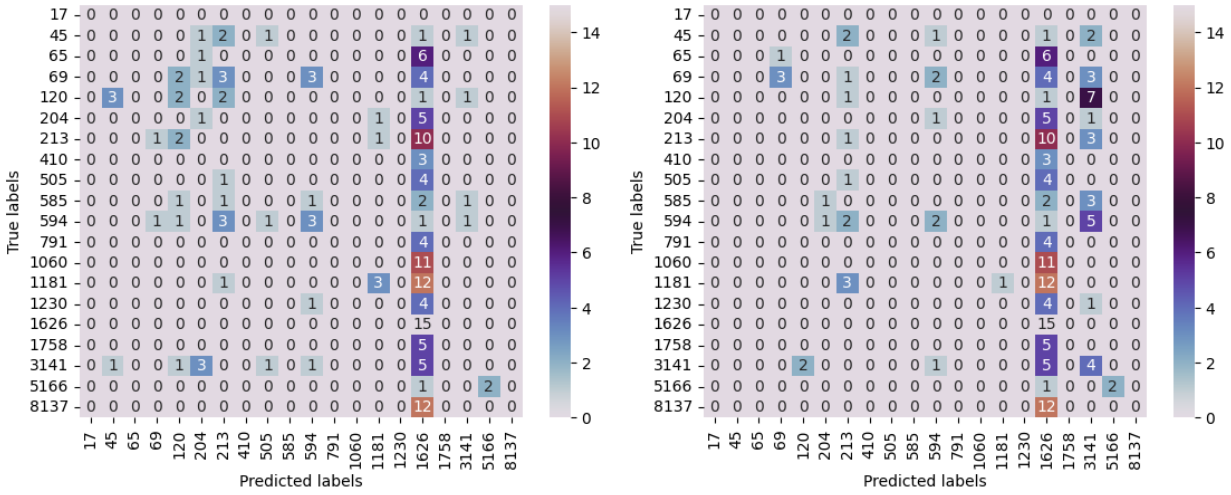
pravdepodobne predučí na jedného autora s veľa dielami a nie je schopný sa naučiť charakteristiky pri menšom počte diel.

7.8 Vyhodnotenie

Počas všetkých pokusov sa ako najviac konzistentné ukázali modely DNN a CNN. Architektúra modelov bola v podstate stále rovnaké, menili sa len počty neurónov v jednotlivých vrstvách. Architektúra pre DNN ja zobrazené v tabuľke 7.12 a pre CNN v tabuľke 7.13. Obidve architektúry sú veľmi jednoduché, zložitejšie architektúry, napríklad s pridaním viacerých vrstiev, podávali horšie výsledky. Keďže datasety neboli veľmi veľké, tieto menšie modely si s nimi vedeli poradiť oveľa lepšie.

Názov vrstvy	**Tvar výstupu**	**Aktivačná funkcia**
Vectorization Layer	m	-
Embedding	sequence_length × embedding_dim	-
GlobalAveragePooling1D	embedding_dim	-
Dense (128)	128	ReLU
Dropout (0.2)	128	-
Dense (64)	64	ReLU
Dropout (0.2)	64	-
Dense (output)	y_train.shape[1]	Softmax

Tabuľka 7.12: Architektúra DNN modelu



(a) (b)

Obr. 7.9: Výsledky LSTM+GRU modelu pri GloVe (a) a FastText(b)

Názov vrstvy	**Tvar výstupu**	**Aktivačná funkcia**
Input Layer	m	-
Vectorization Layer	m	-
Embedding	m × embedding_dim	-
Conv1D	m × 512	ReLU
MaxPooling1D	(m/2) × 512	-
GlobalAveragePooling1D	512	-
Dense	256	ReLU
Dropout (0.2)	256	-
Dense (output)	y_train.shape[1]	Softmax

Tabuľka 7.13: Architektúra CNN modelu

Kapitola 8

Záver

Hlavnou témou diplomovej práce bola identifikácia autorstva v textových dokumentoch. K tomu bola najskôr predstavená potrebná teória. Na začiatok je opísana práca s textovými dátami a preprocessing. Ďalej sú predstavné jednoduché techniky vektorizácie ako One-hot encoding, Bag of Words a TF-IDF, ktoré transformujú dáta do vektorového priestoru. Boli predstavené aj techniky word embeddinigs GloVe a FastText, ktoré dokážu ešte lepšie zachytiť kontext slov a viet a previesť ich do vektorovej podoby. Ďalej boli v teoretickej časti predstavené jednotlivé modely strojového a hlbokého učenia v chronologickom poradí, od najstarších modelov strojového učenia až po najaktuálnejšie modely typu Transformer.

Po oboznamení s potrebnou teóriou nasledovala praktická a implementačná časť. Prvým krokom bolo zíkať dáta, na ktoré bol využitý Projekt Gutenberg. Celkovo bolo vytvorených 6 rôznych datasetov, ktoré sa líšia veľkosťou, počtom autorov, počtom diel a dĺžkou vzatej sekvencie z každého diela. Datasety boli tvorené s ohľadom na konkrétne experimenty.

Po stiahnutí a príprave dát prišla na rad tvorba modelov. Na prvom datasete sa modely tvorili, začalo sa od najjednoduchších a postupne sa pridávali vrstvy a menili hyperparametre. Vybrané modely, ktoré boli ďalej používané na ostatné datasety, mali väčšinou jednoduchú architektúru, pretože tak podávali najlepšie výsledky. Samozrejme ak bolo potreba, boli ešte mierne poupravené pre účely daného datasetu, hlavne počet neurónov v jednotlivých vrstvách.

Pri testovaní modelov sa ukázalo, že najjednoduchšie modely často podajú najpresnejšie výsledky. S metrikou accuracy dosiahla najlepší výsledok v rámci všetkých experimentov hustá neurónová sieť v spojení s Bag of Words vektorizáciou. Táto neurónová sieť a aj model konvolučnej neurónovej siete dosahovali pomerne konzistentné výsledky po celý čas. Napriek tomu, že konvolučná neurónová sieť bola pôvodne vyvinutá na prácu s obrázkami, v mojej práci dosahovala lepšie výsledky ako rekuretná neurónova sieť, ktorá je práve viac pripôsobená na textové dáta.

Modely Transformer, ktoré sú najmodernejšie, nepodávali až také dobré výsledky ako som očakávala. Je možné, že moje datasety neboli pre ne dostatočne veľké. Bohužiaľ z kníh, ktoré boli dostupné z projektu Gutenberg, sa mi nepodarilo zostaviť dataset s veľkým množstvom autorov,

kde by mal aj každý autor dostatok kníh. Práve na väčších dátach sa podarilo dosiahnuť najlepší výsledok za Transformer modely. Bol to model DistilBert v prvom experimente.

Podľa všetkých experimentov by som zhodnotila, že najväčší vplyv na presnosť modelov majú datasety. Najlepšie výsledky väčšinou dosahovali všetky modely pri rovnakých dátach. Za najlepší dataset by som označila jeden z nebalancovaných datasetov, ktorý mal vysoký počet diel pre každého autora, minimálne 58. Samozrejme majú vplyv aj samotné modely, prvé modely pôvodovne dosahovali veľmi nízke výsledky, ale to sa podarilo vylepšiť.

Práca by sa dala ďalej rozšírovať do rôznych smerov. Zaujímavé by mohlo byť otestovať rôzne typy preprocessingu na dátach a následný vplyv na presnosť modelov. Dali by sa vymyslieť aj ďalšie zaujímave experimenty na štýl posledného experimentu, kde sa skúmalo, či si modely budú zamieňať autorov z rôznych období a zistilo sa, že väčšina zámien bola v rámci rovnakého obdobia. Mohli by sa skúmať autori z rôznych častí sveta alebo rôzne vekové kategórie. Tiež by sa ešte oplatilo hlbšie preskúmať, či by sa nedalo pomocou manipulácie s hyperparametrami pri Transformer modeloch vylepšiť presnosť. Niektoré hyperparametre boli odskúšané pri tvorbe tejto práce, ale určite je tam ešte viac možností a kombinácií.

Literatura

1. *What Is Natural Language Processing? | IBM* [online]. 2024-03. [cit. 2024-04-29]. Dostupné z : https://www.ibm.com/topics/natural-language-processing.
2. *Machine learning for text*. 1st edition. New York, NY: Springer Science+Business Media, 2018. ISBN 9783319735306.
3. PIKES, Kurtys. *Stemming and Lemmatization in Python*. 2023-02. Dostupné tiež z: https://www.datacamp.com/tutorial/stemming-lemmatization-python. publisher: datacamp.
4. *5 Types of Word Embeddings and Example NLP Applications* [online]. [B.r.]. [cit. 2024-04-29]. Dostupné z : https://swimm.io/learn/large-language-models/5-types-of-word-embeddings-and-example-nlp-applications.
5. *Bag of Words Model in NLP Explained | Built In* [online]. [B.r.]. [cit. 2024-04-29]. Dostupné z : https://builtin.com/machine-learning/bag-of-words.
6. *TF-IDF — Term Frequency-Inverse Document Frequency – LearnDataSci* [online]. [B.r.]. [cit. 2024-04-29]. Dostupné z : https://www.learndatasci.com/glossary/tf-idf-term-frequency-inverse-document-frequency/.
7. VATSAL. *Word2Vec Explained* [online]. 2023-07. [cit. 2024-04-29]. Dostupné z : https://towardsdatascience.com/word2vec-explained-49c52b4ccb71.
8. VALETI, Dilip. *Classification using Word2vec* [online]. 2021-09. [cit. 2024-04-29]. Dostupné z : https://medium.com/@dilip.voleti/classification-using-word2vec-b1d79d375381.
9. MIKOLOV, Tomas; CHEN, Kai; CORRADO, Greg; DEAN, Jeffrey. *Efficient Estimation of Word Representations in Vector Space* [online]. arXiv, 2013-09 [cit. 2024-04-29]. Dostupné z DOI: 10.48550/arXiv.1301.3781. arXiv:1301.3781 [cs].
10. *GloVe: Global Vectors for Word Representation* [online]. [B.r.]. [cit. 2024-04-29]. Dostupné z : https://nlp.stanford.edu/projects/glove/.
11. DURNA, Merve Bayram. *Advanced Word Embeddings: Word2Vec, GloVe, and FastText* [online]. 2024-01. [cit. 2024-04-29]. Dostupné z : https://medium.com/@mervebdurna/advanced-word-embeddings-word2vec-glove-and-fasttext-26e546ffedbd.

12. WORDPRESS, 2U. *What Is Machine Learning (ML)?* [online]. 2020-06. [cit. 2024-04-29]. Dostupné z : https://ischoolonline.berkeley.edu/blog/what-is-machine-learning/.

13. KRANTIWADMARE. *Logistic Regression in Machine Learning* [online]. 2021-05. [cit. 2024-04-29]. Dostupné z : https://medium.com/analytics-vidhya/logistic-regression-in-machine-learning-f3a90c13bb41.

14. KESELJ, Vlado. Speech and Language Processing (second edition) Daniel Jurafsky and James H. Martin (Stanford University and University of Colorado at Boulder) Pearson Prentice Hall, 2009, xxxi+988 pp; hardbound, ISBN 978-0-13-187321-6, $115.00. *Computational Linguistics* [online]. 2009-09, roč. 35, č. 3, s. 463–466 [cit. 2024-04-29]. ISSN 0891-2017, ISSN 1530-9312. Dostupné z DOI: 10.1162/coli.B09-001.

15. NAZARI, Farnaz; YAN, Wei. *Convolutional versus Dense Neural Networks: Comparing the Two Neural Networks Performance in Predicting Building Operational Energy Use Based on the Building Shape* [online]. arXiv, 2021-08 [cit. 2024-04-29]. Dostupné z DOI: 10.48550/arXiv.2108.12929. arXiv:2108.12929 [cs].

16. AMHERD, Fabian; RODRIGUEZ, Elias. *Heatmap-based Object Detection and Tracking with a Fully Convolutional Neural Network*. 2021-01.

17. *Basic CNN Architecture: Explaining 5 Layers of Convolutional Neural Network* [online]. [B.r.]. [cit. 2024-04-29]. Dostupné z : https://www.upgrad.com/blog/basic-cnn-architecture/.

18. ABUHAMAD, Mohammed; RHIM, Ji-su; ABUHMED, Tamer; ULLAH, Sana; KANG, Sanggil; NYANG, DaeHun. Code authorship identification using convolutional neural networks. *Future Generation Computer Systems* [online]. 2019-06, roč. 95, s. 104–115 [cit. 2024-04-29]. ISSN 0167-739X. Dostupné z DOI: 10.1016/j.future.2018.12.038.

19. YIN, Wenpeng; KANN, Katharina; YU, Mo; SCHÜTZE, Hinrich. *Comparative Study of CNN and RNN for Natural Language Processing* [online]. arXiv, 2017-02 [cit. 2024-04-29]. Dostupné z DOI: 10.48550/arXiv.1702.01923. arXiv:1702.01923 [cs].

20. DURNA, Merve Bayram. *Neural Networks in NLP: RNN, LSTM, and GRU* [online]. 2024-01. [cit. 2024-04-29]. Dostupné z : https://medium.com/@mervebdurna/nlp-with-deep-learning-neural-networks-rnns-lstms-and-gru-3de7289bb4f8.

21. KOSTADINOV, Simeon. *Understanding GRU Networks* [online]. 2019-11. [cit. 2024-04-29]. Dostupné z : https://towardsdatascience.com/understanding-gru-networks-2ef37df6c9be.

22. VASWANI, Ashish; SHAZEER, Noam; PARMAR, Niki; USZKOREIT, Jakob; JONES, Llion; GOMEZ, Aidan N.; KAISER, Lukasz; POLOSUKHIN, Illia. *Attention Is All You Need* [online]. arXiv, 2023-08 [cit. 2024-04-29]. Dostupné z DOI: 10.48550/arXiv.1706.03762. arXiv:1706.03762 [cs].

23. ROTHMAN, Denis. *Transformers for Natural Language Processing Build Innovative Deep Neural Network Architectures for NLP with Python, Pytorch, TensorFlow, BERT, RoBERTa, and More.* Birmingham: Packt Publishing, 2021. ISBN 9781800568631. OCLC: 1236261667.

24. HELENE_K. *Attention and Transformer Models* [online]. 2022-02. [cit. 2024-04-29]. Dostupné z : `https://towardsdatascience.com/attention-and-transformer-models-fe667f958378`.

25. DEVLIN, Jacob; CHANG, Ming-Wei; LEE, Kenton; TOUTANOVA, Kristina. *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding* [online]. arXiv, 2019-05 [cit. 2024-04-29]. Dostupné z DOI: `10.48550/arXiv.1810.04805`. arXiv:1810.04805 [cs].

26. SANH, Victor; DEBUT, Lysandre; CHAUMOND, Julien; WOLF, Thomas. *DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter* [online]. arXiv, 2020-02 [cit. 2024-04-29]. Dostupné z DOI: `10.48550/arXiv.1910.01108`. arXiv:1910.01108 [cs].

27. CLARK, Kevin; LUONG, Minh-Thang; LE, Quoc V.; MANNING, Christopher D. *ELECTRA: Pre-training Text Encoders as Discriminators Rather Than Generators* [online]. arXiv, 2020-03 [cit. 2024-04-29]. Dostupné z DOI: `10.48550/arXiv.2003.10555`. arXiv:2003.10555 [cs].

28. *TensorFlow* [online]. [B.r.]. [cit. 2024-04-29]. Dostupné z : `https://www.tensorflow.org/`.

29. *Keras: The high-level API for TensorFlow | TensorFlow Core* [online]. [B.r.]. [cit. 2024-04-29]. Dostupné z : `https://www.tensorflow.org/guide/keras`.

30. *Project Gutenberg* [online]. [B.r.]. [cit. 2024-04-29]. Dostupné z : `https://www.gutenberg.org/`.

31. PROKOP, Vojtěch. Zpracování textu pomocí hlubokých neuronových sítí [online]. 2022 [cit. 2024-04-29]. Dostupné z : `http://dspace.vsb.cz/handle/10084/147324`.

32. SEOL, Da; CHOI, Jeong; KIM, Chan; HONG, Sang. Alleviating Class-Imbalance Data of Semiconductor Equipment Anomaly Detection Study. *Electronics*. 2023-01, roč. 12, s. 585. Dostupné z DOI: `10.3390/electronics12030585`.

Dodatok A

Kompletné výsledky

Model	Learning rate	Accuracy	Precision	Recall	F1 Score
Bert	5e-5	0,652	0,682	0,668	0,633
Bert	5e-4	0,058	0,042	0,077	0,027
Bert	3e-5	0,611	0,626	0,624	0,586
Bert	8e-5	0,668	0,695	0,684	0,659
DistilBert	5e-5	0,644	0,660	0,664	0,632
DistilBert	5e-4	0,425	0,430	0,434	0,405
DistilBert	3e-5	0,562	0,552	0,587	0,531
DistilBert	8e-5	0,652	0,668	0,665	0,642
Electra	5e-5	0,247	0,232	0,276	0,192
Electra	5e-4	0,112	0,050	0,126	0,059
Electra	3e-5	0,126	0,096	0,131	0,072
Electra	8e-5	0,317	0,289	0,333	0,263

Tabuľka A.1: Výsledky transformer modelov pre rôzne hodnoty learning rate

Model	Accuracy	F1 Score	Precision	Recall
DNN	0,909	0,911	0,922	0,909
CNN	0,921	0,921	0,929	0,921
LSTM+GRU	0,037	0,003	0,001	0,037
Bidirectional LSTM	0,876	0,874	0,887	0,878
LSTM	0,083	0,062	0,102	0,083

Tabuľka A.2: Výsledky modelov s väčšou veľkosťou vektorizácie

Model	Accuracy	F1 Score	Precision	Recall
DNN	0,897	0,899	0,911	0,897
CNN	0,926	0,926	0,936	0,926
LSTM+GRU	0,488	0,477	0,552	0,488
Bidirectional LSTM	0,636	0,636	0,721	0,636
LSTM	0,107	0,076	0,088	0,107

Tabuľka A.3: Výsledky modelov s menšou veľkosťou vektorizácie

Model	Vektorizácia	Čas (s)	Accuracy	F1 Score	Precision	Recall
DNN	TF-IDF	269	0,702	0,702	0,766	0,702
CNN	TF-IDF	201	0,702	0,696	0,741	0,702
LSTM+GRU	TF-IDF	1147	0,048	0,004	0,002	0,048
Bidirectional LSTM	TF-IDF	933	0,511	0,517	0,574	0,511
DNN	GloVe	1902	0,705	0,687	0,720	0,705
CNN	GloVe	1866	0,665	0,622	0,638	0,664
LSTM+GRU	GloVe	1042	0,142	0,110	0,119	0,142
Bidirectional LSTM	GloVe	1914	0,651	0,623	0,651	0,651
DNN	FastText	1286	0,608	0,576	0,586	0,608
CNN	FastText	1065	0,608	0,578	0,589	0,608
LSTM+GRU	FastText	671	0,139	0,080	0,058	0,139
Bidirectional LSTM	FastText	1224	0,474	0,415	0,406	0,474

Tabuľka A.4: Výsledky a čas trénovania pre modely neurónových sietí

Model	Vektorizácia	Accuracy	F1 Score	Precision	Recall	Čas (s)
Log.Regresia	BagOfWords	0,860	0,829	0,818	0,860	830
NaiveBayes	BagOfWords	0,494	0,409	0,448	0,494	7
Log.Regresia	TF-IDF	0,506	0,433	0,477	0,506	188
NaiveBayes	TF-IDF	0,186	0,117	0,115	0,186	5
Log.Regresia	GloVe	0,405	0,288	0,266	0,406	5
NaiveBayes	GloVe	0,496	0,420	0,417	0,496	<1
Log.Regresia	Word2Vec	0,251	0,132	0,115	0,251	4
NaiveBayes	Word2Vec	0,486	0,408	0,402	0,486	<1
Log.Regresia	Bert	0,517	0,466	0,486	0,517	45
NaiveBayes	Bert	0,424	0,362	0,368	0,424	<1
Log.Regresia	Distilbert	0,478	0,402	0,394	0,478	39
NaiveBayes	Distilbert	0,416	0,356	0,353	0,416	<1
Log.Regresia	Electra	0,416	0,351	0,331	0,416	12
NaiveBayes	Electra	0,264	0,196	0,239	0,264	<1

Tabuľka A.5: Výsledky a čas trénovania pre modely strojového učenia

Model	**Vektorizácia**	**Accuracy**	**F1 Score**	**Precision**	**Recall**
DNN	TF-IDF	0,805	0,790	0,801	0,805
CNN	TF-IDF	0,829	0,834	0,861	0,829
Bidirectional LSTM	TF-IDF	0,689	0,678	0,707	0,689
DNN	GloVe	0,713	0,698	0,758	0,713
CNN	GloVe	0,787	0,776	0,808	0,787
Bidirectional LSTM	GloVe	0,780	0,755	0,784	0,780
DNN	FastText	0,695	0,651	0,728	0,695
CNN	FastText	0,707	0,673	0,718	0,707
Bidirectional LSTM	FastText	0,512	0,460	0,487	0,512

Tabuľka A.6: Výsledky pre autorov z rôznych storočí