VŠB TECHNICKÁ UNIVERZITA OSTRAVA
FAKULTA ELEKTROTECHNIKY A INFORMATIKY

# CFP Search: vyhledávání v Call for Papers

CFP Search: search in Call for Papers

Bc. Marek Hanuš

Diplomová práce

Vedoucí práce: doc. Ing. Pavel Krömer, Ph.D.

Ostrava, 2023

VŠB TECHNICKÁ UNIVERZITA OSTRAVA | FAKULTA ELEKTROTECHNIKY A INFORMATIKY

Č.j.VSB/22/078917
vsboee22025FEB

# Zadání diplomové práce

Student: **Bc. Marek Hanuš**

Studijní program: N2647 Informační a komunikační technologie

Studijní obor: 2612T025 Informatika a výpočetní technika

Téma: CFP Search: vyhledávání v Call for Papers
CFP Search: search in Call for Papers

Jazyk vypracování: čeština

Zásady pro vypracování:

WikiCFP a DBWorld jsou populární služby pro online sdílení anoncí vědeckých akcí (konferencí, workshopů, letních škol atd.). Jejich schopnosti vyhledávání jsou ale značně omezené a nepodporují pokročilé funkce jako např. kontextová doporučení. Cílem práce je analyzovat textová CFP data a vytvořit vyhledávací aplikaci s podporou automatické klasifikace a doporučení relevantních konferencí.

Řešení bude zahrnovat následující kroky:
1. Studium a přehled metod pro automatickou klasifikaci textu.
2. Implementaci vybraných metod.
3. Testování na datech z WikiCFP, DBWorld nebo dalších vhodných zdrojích.
4. Analýzu, vizualizaci a zhodnocení dosažených výsledků.

Seznam doporučené odborné literatury:

[1] Jochen Hartmann, Juliana Huppertz, Christina Schamp, Mark Heitmann, Comparing automated text classification methods, International Journal of Research in Marketing, Volume 36, Issue 1, 2019, Pages 20-38, ISSN 0167-8116, https://doi.org/10.1016/j.ijresmar.2018.09.009.

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

Vedoucí diplomové práce: **doc. Ing. Pavel Krömer, Ph.D.**

Datum zadání: 01.09.2022

Datum odevzdání: 30.04.2023

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

V IS EDISON zadáno: 07.11.2022 11:59:21

## Abstrakt

Tato diplomová práce se zabývá zpracováním přirozeného jazyka za účelem klasifikace textových dokumentů. Pro trénování klasifikátorů byly zvoleny datové kolekce získané ze stránek WikiCFP a DBWorld. Úvod práce se zabývá získáním těchto dat, jejich základní analýzou a předzpracováním za účelem vytvoření očištěné datové sady. Následuje podrobný popis metod pro vytvoření vektorové reprezentace textu a přehled vhodných modelů pro automatickou klasifikaci, a to od klasických přístupů, přes neuronové sítě až po aktuální state-of-the-art techniky. Vybrané modely z jednotlivých kategorií byly implementovány a otestovány nad specifikovanými datovými zdroji. Výsledky jsou podrobně analyzovány, vizualizovány a zhodnoceny. Vytvořené modely byly implementovány do praktické webové aplikace.

## Klíčová slova

klasifikace textu; zpracování přirozeného jazyka; strojové učení; neuronové sítě; hluboké učení; webová aplikace; anonce konferencí; WikiCFP; DBWorld

## Abstract

This thesis focuses on natural language processing for classification of text documents. For classifier training were selected data collections obtained from WikiCFP and DBWorld. The introduction of the thesis deals with the acquisition of these data, their basic analysis and preprocessing to create a clean dataset. This is followed by a detailed description of the methods for creating a vector representation of the text and an overview of suitable models for automatic classification, ranging from classical approaches, through neural networks to current state-of-the-art techniques. Selected models from each category have been implemented and tested over specified data sources. The next part of the work deals with testing on specified data sources. Results are analysed, visualised, and evaluated. Created models have been implemented to practical web application.

## Keywords

text classification; natural language processing; machine learning; neural networks; artificial neural networks; deep learning; web application; Call for Papers; WikiCFP; DBWorld

## Poděkování

Rád bych poděkoval svému vedoucímu této práce, kterým je doc. Ing. Pavel Krömer, Ph.D. a to za odborné vedení, konzultace a promptní komunikaci při zpracování práce. Za cenné tipy k práci také děkuji Bc. Jakubu Kólovi. Na závěr děkuji své rodině a blízkým za trpělivost a podporu při finalizaci této práce.

# Obsah

# Seznam použitých zkratek a symbolů

| | | |
|---|---|---|
| ANN | – | Artificial neural network |
| API | – | Application programming interface |
| BPE | – | Byte Pair Encoding |
| BSON | – | Binary JSON |
| BoW | – | Bag-of-words |
| CBOW | – | Continuous bag-of-words |
| CFP | – | Calls For Papers |
| CNN | – | Convolutional neural network |
| CPU | – | Central processing unit |
| CSS | – | Cascading Style Sheets |
| CSV | – | Comma-Separated Values |
| CUDA | – | Compute Unified Device Architecture |
| DL | – | Deep learning |
| DOM | – | Document Object Model |
| GBM | – | Gradient Boosting Machine |
| GPU | – | Graphics processing unit |
| GRU | – | Gated recurrent unit |
| GloVe | – | Global Vectors |
| HTML | – | HyperText Markup Language |
| HTTP | – | Hypertext Transfer Protocol |
| IDE | – | Integrated development environment |
| JSON | – | JavaScript Object Notation |
| KNN | – | K-nearest neighbors |
| LSTM | – | Long short-term memory |
| ML | – | Machine learning |
| MVP | – | Minimum viable product |
| NER | – | Named-entity recognition |
| NLP | – | Natural language processing |

| | | |
|---|---|---|
| NLTK | – | Natural Language Toolkit |
| NoSQL | – | non-SQL |
| POC | – | Proof of concept |
| POS | – | Part of speech |
| RAM | – | Random-access memory |
| RF | – | Random forest |
| RNN | – | Recurrent neural network |
| SIMD | – | Single instruction, multiple data |
| SQL | – | Structured Query Language |
| SVM | – | Support vector machines |
| TF-IDF | – | Term Frequency-Inverse Document Frequency |
| TPU | – | Tensor Processing Unit |
| VRAM | – | Video RAM |
| XML | – | Extensible Markup Language |

# Seznam obrázků

# Seznam tabulek

# Kapitola 1

# Úvod

Cílem této práce je zpracovat rešerši metod pro automatickou klasifikaci textu. Na začátku bude nutné jednotlivé metody detailně nastudovat a stručně popsat. Z množství existujících algoritmů je dále nutné vybrat vhodné metody k implementaci a aplikovat je nad daty WikiCFP, případně DBWorld. Veškeré experimenty je nutné vyhodnotit pomocí evaluačních metrik, které zahrnují jak číselné hodnoty vyjadřující výkonnost modelu, tak i grafické vizualizace. Nedílnou součástí je objektivní zhodnocení dosažených výsledků, případně odůvodnění jejich neúspěchu. Závěrem zpracování je také vytvoření vyhledávací aplikace, která bude demonstrovat využití klasifikace a umožní uživateli doporučit relevantní konference.

Téma práce lze zařadit do oboru zpracování přirozeného jazyka neboli *natural language processing*. Klasifikací je ve statistice označován proces, ve kterém jsou, dle vypozorovaných charakteristik, seskupovány související entity do tříd. Každý objekt spadá právě do jedné ze tříd, přičemž tyto třídy jsou vzájemně disjunktní a nemohou se překrývat.

Call for Papers je označení pro výzvu k předkládání příspěvků do konference, workshopu, časopisu nebo jiné akce. CFP většinou zahrnuje informace o tématu akce, požadavky na předkládané příspěvky (např. délka, formát, jazyk), termíny pro podání příspěvků a další důležité informace pro účastníky. CFP bývá publikováno na webových stránkách dané akce, v odborných časopisech nebo prostřednictvím sociálních sítí a e-mailových seznamů. Předkládání příspěvků na základě CFP je důležitou součástí akademického procesu a umožňuje vědcům a specialistům prezentovat své výzkumné výsledky a diskutovat o nich s kolegy z oboru.

V první kapitole uvedu čtenáře do dostupných zdrojů obsahujících Call for Papers záznamy, včetně uvedení základních parametrů a porovnání s jinými nalezenými zdroji. Použité datasety obsahují i popis metod k jejich získání a postup implementace pomocných stahovacích skriptů.

Na získané datasety navazuje i předzpracování dat pro využití v klasifikačních metodách. Uvedené zpracování kategorií se stalo poměrně komplikovanou problematikou. Jako součást předzpracování se dá považovat i převod textu do vektorových reprezentací, se kterými jsou schopny pracovat klasifikační algoritmy.

V páté kapitole se čtenář seznámí se základními pojmy a obecně problematikou klasifikace. Na téma navazuji přehledem klasifikačních metod. Každý algoritmus obsahuje alespoň základní popis, přičemž metody důležité pro tuto práci jsou vysvětleny podrobněji. Na klasifikaci navazuje i kapitola zabývající se způsoby vyhodnocení kvality modelů.

Postup implementace vybraných metod popisuji v osmé kapitole. S těmito metodami je detailněji experimentováno v přiložených souborech, ale jsou zde vybrány zajímavé vizualizace a některá zhodnocení dosažených výsledků. Navazující kapitola zachycuje tvorbu jednoduché webové aplikace pro praktické využití implementovaných metod.

Poslední část práce se věnuje popisu využitých technologií, a to jak ze softwarové, tak i hardwarové části. Jen díky množství knihoven a dostupných výpočetních prostředků bylo možné provést velké množství experimentů a zpracovat tuto práci.

# Kapitola 2

# Datové sady

Datovou sadou se rozumí soubor textových dokumentů, které jsou opatřeny štítky, anglicky *labels*, označující třídu, do které záznam patří. Textové dokumenty se mohou skládat z nadpisů, popisu, doplňujících informací včetně odpovídajícího štítku třídy. Jednotlivé atributy mohou být různého datového typu a také libovolné délky, jazyka apod. Všechna nebo vybraná data jsou po předzpracování poté využita k trénování a evaluaci ML modelů. Výkonnost modelů bývá závislá na množství dat. Kvalitu dat lze obecně považovat za klíčovou vlastnost, neboť jedině z kvalitních dat lze naučit přesný model.

## 2.1 WikiCFP

WikiCFP je webová platforma, která sbírá informace o konferencích, workshopech a dalších akcích po celém světě v oblasti počítačových věd a informačních technologií. Tato platforma je určena jak pro výzkumníky a studenty, kteří hledají příležitosti ke zveřejnění svých prací, tak i pro organizátory akcí, kteří chtějí propagovat své akce a získat více účastníků.

Na WikiCFP je možné vyhledávat akce podle tématu, termínu, místa konání a dalších kritérií. U každé akce jsou zobrazeny podrobné informace, jako jsou názvy a popisy přednášek, termíny odevzdání příspěvků, místo konání, registrační poplatky a další.

### 2.1.1 Získání dat

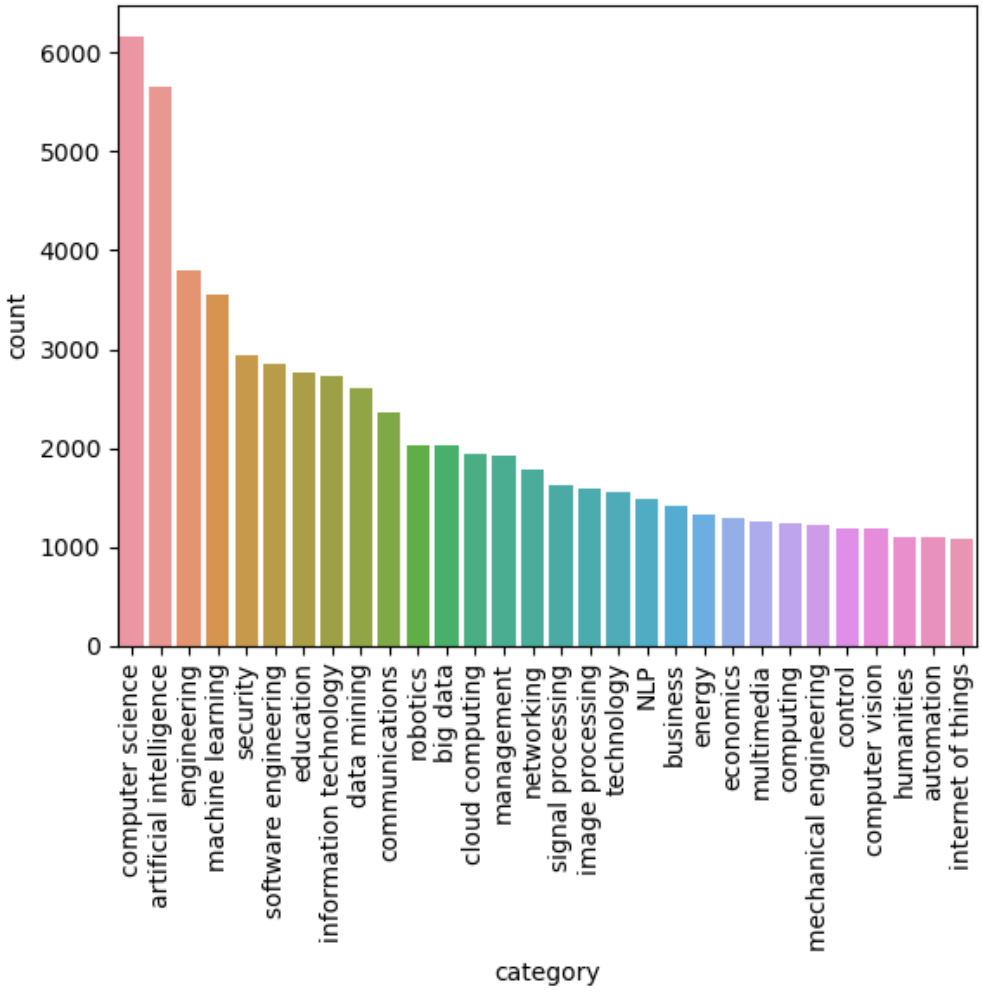
Nejprve byla provedena rešerše dostupných možností pro stažení dat. Stránka neposkytuje API rozhraní a generované XML exporty jsou již několik let neaktualizované.

Proto jsem se pokusil najít, zda existuje již implementovaný skript ke stažení CFP dat. Bohužel ani jeden z dostupných nástrojů nevyhovoval mým potřebám. Některé skripty byly v jazyce Haskell, některé byly napsány ještě pro Python verze 2.

Pro získání celé datové sady byl tedy implementován skript, který postupně stahoval jednotlivé stránky. Dle podmínek WikiCFP musely být pauzy mezi jednotlivými požadavky na server minimálně 5 sekund.

### 2.1.2 Explorační analýza

Celkově data WikiCFP obsahují 94270 záznamů při spotřebě cca 4 GB místa na disku. Záznamy se skládají ze základních informací jako jsou název konference, podrobný popis, místo konání a odkaz na web. Každý záznam může obsahovat až 4 třídy. Celkově dataset zahrnuje desetitisíce tříd, ze kterých jsem vybral 30 nejpočetnějších do vizualizace jejich distribuce (obrázek 2.1).




Obrázek 2.1: Vizualizace distribuce 30 nejpočetnějších tříd

## 2.2 DBWorld

DBWorld je webová stránka zaměřená na databázové technologie a obecně na oblast zpracování dat. Poskytuje informace o konferencích, workshopech, výzvách a oznámeních v oblasti databázových technologií a souvisejících oblastí, jako jsou například data mining, strojové učení, velká data, umělá inteligence a další. DBWorld poskytuje aktualizovaný seznam příspěvků, přičemž nové záznamy jsou běžně přidávány a staré jsou pravidelně aktualizovány. Tato stránka je užitečným zdrojem informací pro výzkumníky a odborníky v oblasti databázových technologií a souvisejících oblastí, kteří chtějí zůstat informováni o nejnovějších výzkumných aktivitách a vývoji v této oblasti.

Uvedený mailing list poskytuje celkem 3 typy RSS kanálů. Bohužel vzhledem k neexistujícím třídám dataset nemohl být dále využit pro trénování a testování modelů.

## 2.3 Shrnutí

Vzhledem k nedostatečné kvalitě stávajících zdrojů jsem provedl průzkum dalších online dostupných databází. Ani jedna z uvedených v tabulce 2.1 neodpovídala požadavkům na množství dat a kvalitní anotaci kategorií.

Tabulka 2.1: Porovnání databází obsahující Call for Papers

| Název databáze | Přibližný počet příspěvků |
|---|---:|
| WikiCFP[1] | 155 000 |
| DBWorld[2] | 4 000 |
| IEEE Conferences[3] | 6 878 |
| ACM Conferences[4] | 2 800 |
| Conference Alerts[5] | 9 000 |
| dblp[6] | 13 010 |
| All Conference CFP Alerts[7] | 19 000 |

[1] http://www.wikicfp.com/cfp/
[2] https://dbworld.sigmod.org/browse.html
[3] https://conferences.ieee.org/conferences_events/
[4] https://dl.acm.org/proceedings; https://dl.acm.org/icps
[5] https://www.conferencealerts.org/
[6] https://dblp.org/
[7] https://allconferencecfpalerts.com/

# Kapitola 3

# Předzpracování datových sad

Preprocessing dat je důležitý krok v přípravě dat pro strojové učení. Cílem preprocessingových úprav je zajistit, aby data byla vhodná pro modely strojového učení a aby modely mohly co nejlépe pracovat s těmito daty.

Textové popisy konferencí ve WikiCFP jsou relativně standardním textem v anglickém jazyce, a proto se zde předzpracování nijak výrazně neliší od běžných technik.

## 3.1 Textová data

### 3.1.1 Převod na malá písmena

Tento krok slouží k tomu, aby se snížila variabilita textu a zajistilo se, že stejné slovo napsané různým způsobem (např. s velkým a malým počátečním písmenem) bude interpretováno jako stejné slovo.

### 3.1.2 Stop slova

Stop words jsou slova, která se vyskytují velmi často v textu a nemají pro analýzu přínos. Příkladem mohou být spojky (např. „a“, „i“, „nebo“), zájmena (např. „on“, „ona“, „oni“) nebo často používaná slovesa (např. „být“, „mít“). Odstranění stop slov může zlepšit výsledky analýzy textu, protože se zaměříme na významově bohatší slova. Při odstraňování stop slov v preprocessingu se tyto slova jednoduše odstraní z textu, aby se snížila zátěž na výpočetní zdroje a zlepšila se kvalita dat.

### 3.1.3 Stematizace

Stematizace je proces, kdy se slova redukují na jejich kořenovou formu, takže slova se stejným kořenem jsou shledána jako stejná. To zahrnuje odstranění přípon, koncovek a jiných morfologických prvků slov, které se často neberou v úvahu při analýze textu. Například slova „běhám“, „běžíme“, „běhání“ by byla převedena na kořenové slovo „běh“. Stemming je často používán jako krok před

zavedením dalších metod, jako je vektorové reprezentování textu, což umožňuje redukovat dimenzi dat a tím zlepšit výkon modelu.

### 3.1.4 Lemmatizace

Lemmatizace je proces úpravy slov na jejich základní tvar, tzv. lemma. Jedná se o výrazovou úpravu, kdy se slova převádějí na základní tvar za účelem zjednodušení textových dat a snížení dimenze prostoru při analýze. Například slova jako „kočka“, „kočce“ a „kočku“ by mohla být převedena na stejnou základní formu „kočka“.

### 3.1.5 Odstranění irelevantních informací

**Internetové adresy** – Jedná se o odkazy na webové stránky, které obvykle nepřinášejí přidanou hodnotu pro klasifikaci.

**E-mailové adresy** – Jsou to kontaktní informace, které nejsou relevantní pro textovou klasifikaci.

**Telefonní čísla** – Stejně jako v předchozím případě se jedná o irelevantní kontaktní informace.

**Kalendářní data** – Označení data nebo období, které většinou popisuje deadline pro odevzdání příspěvků. Označení času se v CFP ve větší míře nevyskytuje.

**Interpunkce** – Symboly interpunkce jsou často používány pro gramatiku a sémantiku věty.

**Speciální znaky** – Speciální znaky jsou znaky, které se nevyskytují v běžných slovech, což mohou být kromě interpunkčních znamének, například číslice, emotikony, nebo jiné symboly.

## 3.2 Kategorická data

### 3.2.1 Odstranění duplicit a optimalizace kategorií

Redukce duplicitních kategorií je součástí preprocessingu, který pomáhá zlepšit kvalitu a přesnost textových modelů. Jedním z problémů můžou být kategorie, které mají stejný či podobný význam, ale různé názvy. To vede k rozdílnému označení stejných entit a snížení přesnosti modelu.

Příkladem uvedeného problému jsou data WikiCFP, které obsahují tisíce kategorií. Prvotní pokusy spočívaly ve výběru podmnožiny relevantních kategorií bez slučování. Množství trénovacích dat bylo touto metodou omezeno na řádově tisíce záznamů, z toho desítky až stovky v jednotlivých kategoriích, což vedlo k explozi gradientu při učení modelu. Proto bylo v tomto případě přistoupeno k redukci duplicitních kategorií. To obnáší manuální vyhledání a sloučení kategorií, které mají stejný význam, ale liší se pouze v názvu. Navazující fází je sloučení kategorií, které mají odlišný význam, ale týkají se stejného tématu.

Například uvedené kategorie „machine learning“, „artificial intelligence“, „neural networks“, „deep learning“ a „computer vision“ mohou být sloučeny do jedné kategorie s názvem „machine learning“. Celkově bylo uvedenými metodami postupně optimalizováno 300 nejčastějších a relevantních kategorií do 10 skupin. Každá skupina reprezentovala pro klasifikátor jednu třídu. Zbylé kategorie obsahovaly řádově desítky CFP a jejich optimalizace nevedly k zásadnímu zlepšení modelů.

# Kapitola 4

# Reprezentace textových dat

Pro další zpracování je potřeba převést text na jinou formu, kterou lze použít pro algoritmy strojového učení. Vytvořený vektor se nazývá *embedding*. Existuje množství technik vektorizace. Ty si můžeme rozdělit do základních skupin jako vektorizace slov (kapitola 4.1), větších celků jako jsou celé dokumenty (kapitola 4.2) a samostatně algoritmy pro transformer modely (kapitola 4.3).

## 4.1 Vektorizace slov

Word embeddings jsou reprezentace slov z textu v podobě vektorů v n-rozměrném prostoru, kde vzdálenost mezi vektory odpovídá vzdálenosti mezi slovy v jejich sémantickém a syntaktickém významu.

### 4.1.1 Bag-of-words

Bag-of-words (BoW) [1] je jedna z nejjednodušších metod reprezentace textu. Algoritmus předpokládá, že význam textu může být získán z informace o výskytu jednotlivých slov v textu bez ohledu na jejich pořadí.

Pro reprezentaci textu pomocí bag-of-words se nejprve vytvoří slovník obsahující všechna slova, která se vyskytují v trénovacích datech. Poté se každému slovu přiřadí jedinečné číslo (index v tomto slovníku).

Pro každý text se poté vytvoří vektor o délce slovníku, kde každý prvek vektoru odpovídá jednomu slovu ze slovníku. Hodnota prvku je rovna počtu výskytů odpovídajícího slova v textu. Pokud se slovo v textu nevyskytuje, je jeho počet výskytů vektoru roven nule.

### 4.1.2 TF-IDF

TF-IDF [1] přístup je podobný BoW, ale snaží se upřednostnit slova (termy $t$), která jsou v dokumentu ($d$) důležitější. Frekvence slov ($tf$) se násobí převrácenou frekvencí dokumentu ($idf$), což

umožňuje, aby méně častá slova v dokumentu měla větší váhu. Inverzní frekvence dokumentu se definuje pomocí celkového počtu dokumentů ($N$) a počtu dokumentů, ve kterých se slovo vyskytuje ($df$).

$$idf_t = log\frac{N}{df_t} \tag{4.1}$$

$$tf\text{-}idf_{t,d} = tf_{t,d} \times idf_t \tag{4.2}$$

### 4.1.3 Word2Vec

Word2Vec [2] je algoritmus pro vytvoření word embeddings, který je založen na predikci slov v kontextu. Konkrétně existují dva typy architektur, continuous bag-of-words (CBOW) a skip-gram. U obou se pracuje s okny slov v kontextu, z nichž se snažíme na základě jednoho slova předpovědět ostatní slova v okolí (CBOW) nebo naopak, na základě kontextu předpovídat jedno slovo (skip-gram). Výsledkem těchto architektur je vektorová reprezentace slov, ve které jsou slova s podobným významem reprezentována blízkými vektory. Tato reprezentace slov umožňuje provádět různé úlohy, jako například hledání slov s podobným významem, řešení analogických úloh nebo klasifikaci textu.

### 4.1.4 GloVe

GloVe (Global Vectors) [2] je algoritmus pro vytváření vektorových reprezentací slov (word embeddings), který kombinuje dvě myšlenky: prostorovou reprezentaci slov založenou na distribuční hypotéze a maticovou faktorizaci.

### 4.1.5 fastText

FastText [2] je open-source knihovna pro tvorbu word embeddings, vyvinutá výzkumníky z Facebook AI Research. Jedná se o rozšíření metody Word2Vec, která umožňuje zahrnout do výpočtu nejen slova, ale i významové celky menší než slova, jako jsou například morfémy nebo n-gramy.

## 4.2 Vektorizace dokumentů

Document embeddings jsou reprezentace celých dokumentů, které se používají pro různé úlohy, jako je kategorizace dokumentů, clustering dokumentů, porozumění dokumentům a další.

Podobně jako word embeddings, kde jsou slova reprezentována jako vektory, jsou document embeddings odvozeny z jednotlivých slov dokumentu nebo z celého dokumentu samotného. Reprezentovány jsou jako vektory s nízkou dimenzí, odvozené z distribuce slov v dokumentu a vztahů mezi slovy.

### 4.2.1 Sentence2Vec

Sentence2Vec je technika pro vytváření vektorových reprezentací pro celé věty nebo větší textové bloky. Oproti bag-of-words a TF-IDF reprezentacím, které zachycují pouze počet výskytů slov v textu, umožňují sentence embeddings zohlednit také význam slov a kontext, ve kterém se vyskytují.

### 4.2.2 Doc2Vec

Doc2Vec, známý také jako paragraph2vec nebo sentence embeddings, je metoda pro převod celých dokumentů na vektory nízké dimenze, která se zakládá na algoritmu zvaném Distributed Memory Model of Paragraph Vectors (PV-DM). Podobně jako u Word2Vec algoritmu, Doc2Vec využívá umělé neuronové sítě k učení vektorových reprezentací slov, ale v případě Doc2Vec jsou vytvářeny vektorové reprezentace nejen pro slova, ale i pro celé dokumenty.

## 4.3 Vektorizace pro transformer modely

Pro transformer modely se obvykle používá tzv. subword tokenizace, která pracuje s podřetězci slov a umožňuje zachytit různé tvary slov a slovních tvarů. Subword tokenizace se provádí pomocí algoritmů jako jsou Byte Pair Encoding (BPE) nebo SentencePiece.

### 4.3.1 BPE

Byte Pair Encoding (BPE) je algoritmus pro kompresi dat, který byl později upraven pro použití v tokenizaci pro modely zpracování přirozeného jazyka, zejména pro transformer modely.

### 4.3.2 WordPiece

WordPiece je založen na principu postupného rozdělování slov na menší jednotky (subword units). Algoritmus nejprve rozdělí vstupní text na jednotlivá slova a následně je rozdělí na ještě menší jednotky, které se nazývají subwords. Tyto subwords mohou být samostatná slova, ale také jejich části, které se vyskytují v různých slovech. Například se může stát, že se v datasetu vyskytuje slovo „running“, ale také „run“ a „ning“. Tyto části slov pak mohou být rozděleny do samostatných subwords a tyto subwords se použijí pro představení slova vektorovým způsobem.

### 4.3.3 SentencePiece

SentencePiece je algoritmus pro tokenizaci textu, který byl vyvinutý společností Google a vydán jako open-source knihovna. SentencePiece používá unigramový model, což znamená, že rozdělení textu na jednotlivé tokeny je založeno na četnosti výskytu jednotlivých slov a podslov v trénovacích datech. V praxi to znamená, že nejčastější slova budou tvořit samostatné tokeny, zatímco méně častá slova budou rozdělena na menší podtokeny.

# Kapitola 5

# Klasifikace

## 5.1 Klasifikace, kategorizace, shlukování

Ve statistice je klasifikace problémem zařazení pozorování do sady kategorií na základě vypozorovaných charakteristik. Algoritmus, který implementuje klasifikaci se nazývá klasifikátor a je možné si jej představit jako matematickou funkci, která mapuje vstupní data do kategorií. Vstupní data někdy mohou být nazývány jako regresory, v oboru strojového učení také jako vektory vlastností, anglicky jako *feature vectors*. Predikované výstupní kategorie jsou nazývány jako třídy, anglicky *classes*.

Problém klasifikace je nutné oddělit od kategorizace, která je procesem dělení vstupních dat do skupin, jejíž entity jsou si nějakým způsobem podobné. Zásadní rozdílem oproti klasifikaci je, že tyto skupiny nemusí být vzájemně disjunktní, a tedy jedna entita může být zařazena do více skupin.

Další variantou je shlukování, anglicky *clustering*. Jedná se o úlohu na seskupování množiny objektů takovým způsobem, že objekty v rámci skupiny si jsou určitým způsobem podobnější než objekty v jiných skupinách. Zásadním rozdílem oproti klasifikaci je, že shlukování patří mezi metody bez učitele, tedy nepoužívají trénovací data a shluky zůstávají nepojmenované.

V současné době je klasifikace textů řešena nejčastěji pomocí modelů využívající slovní prvky, např. bag-of-words, n-grams nebo word-embeddings. Jako klasifikační algoritmus lze použít TF-IDF, K-means, CNN (konvoluční neuronové sítě) ale i LSTM.

Všechny metody nicméně vyžadují předzpracování textu. Tento úkol zahrnuje odstranění stop-words, sjednocení na malá/velká písmena, tokenizace vět na slova a lemmatizace nebo stemming jednotlivých slov (tokenů). Např. i pro lemmatizaci a stemming je možné využít samostatné neuronové sítě.

Následuje výpočet vektorových reprezentací, pro který je nutné využít specializované algoritmy. Za nejpopulárnější modelovou architekturu pro výpočet vektorových reprezentací je možno považovat např. Word2Vec. Tento model se využívá k učení slovních asociací z velkého korpusu textu. Po natrénování je možnost detekovat např. synonyma. Výsledná vektorová reprezentace by poté

měla mít možnost pomocí matematické funkce (např. kosinové podobnosti) udávat sémantickou podobnost mezi tokeny.

Stejnou myšlenku jako u Word2Vec lze použít i pro věty (Sentence2Vec) nebo celé dokumenty (Doc2Vec).

Alternativním modelem může být GloVe, který se využívá pro distribuovanou reprezentaci slov. Model se dá zahrnout oproti předchozím do kategorie učení bez učitele. Tento algoritmus nicméně není vhodný pro identifikaci homografů, tedy slov, která se shodně píší, ale mají odlišný význam.

# Kapitola 6

# Klasifikační metody

V této kapitole postupně popíšu metody pro klasifikaci textu se zaměřením na způsob učení s učitelem. To znamená, že disponujeme předem anotovanými daty, která jsou v tomto případě rozdělena do více tříd. Z tohoto důvodu nebude možné využít samostatně metody určené pro binární klasifikaci.

Na začátku uvedu klasické modely, mezi které patří logistická regrese (kapitola 6.1), rozhodovací stromy (kapitola 6.2), dále pokročilejší kombinace stromů pomocí ensemble technik (kapitola 6.3), pravděpodobnostní klasifikátory (kapitola 6.4), metodu nejbližších sousedů (kapitola 6.5) a dělení prostoru pomocí vektorů (kapitola 6.6). V druhé části rozeberu detailněji neuronové sítě (kapitola 6.7), včetně popisu konvolučních (kapitola 6.8) a rekurentních vrstev (kapitola 6.9), dále autoenkodéry (kapitola 6.10) a závěrem i aktuální state-of-the-art transformer modely (kapitola 6.11). Uvedené metody je možné využít nejen na klasifikaci CFP do kategorií, ale také k analýze sentimentu, filtrování spamu, modelování témat a v dalších úlohách.

## 6.1 Logistická regrese

Logistická regrese [3] z řady lineárních modelů je základní statistická metoda používaná pro klasifikaci. Hlavním cílem je odhad pravděpodobnosti, že vstupní data patří do určité třídy. Využívá lineární regresi pro predikci, ale výsledkem je pravděpodobnostní hodnota namísto konkrétního výstupu. Výstup logistické regrese se pohybuje v rozmezí 0 až 1, což lze interpretovat jako pravděpodobnost, že vstupní data patří do jedné ze tříd.

Algoritmus se snaží nalézt optimální sadu vah. To se provádí pomocí procesu zvaného odhad pravděpodobnosti výskytu zkoumaného jevu, který zahrnuje proces iterativní úpravy vah, dokud model nevytvoří nejvyšší možnou pravděpodobnost výskytu zkoumaného jevu.

Podle typu použití dělíme logistickou regresi na binární, multinomickou (kapitola 6.1.1) a ordinální regresi (kapitola 6.1.2).

### 6.1.1 Multinomická logistická regrese

Kromě klasické binární regrese může být logistická regrese použita také pro více tříd. Metoda se pak nazývá nominální logistická regrese nebo také multinomická logistická regrese. V takovém případě model obsahuje více lineárních rovnic a výstupní hodnota se interpretuje jako pravděpodobnost zatřídění do jedné ze tříd.

### 6.1.2 Ordinální logistická regrese

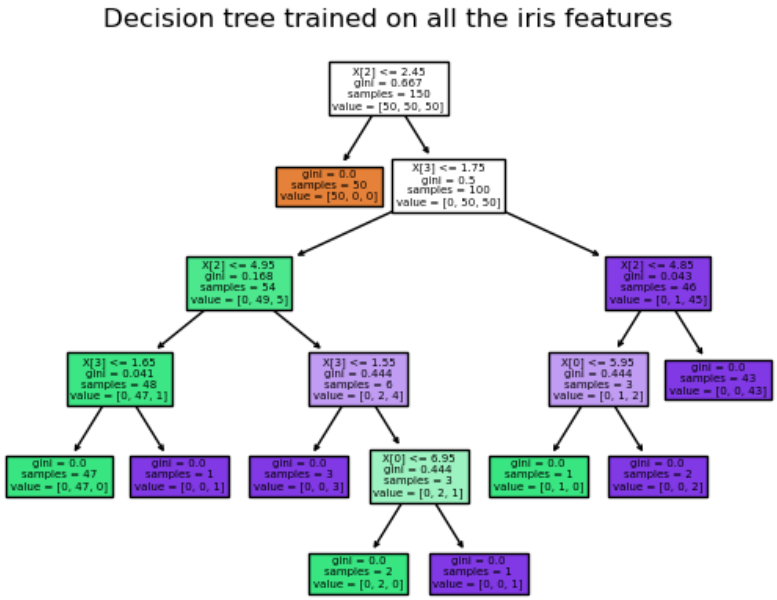
Ordinální logistická regrese je variantou logistické regrese, která se využívá pro klasifikaci vícerozměrných dat, kde výstupem je uspořádaná kategorie (tj. třídy existují v určitém pořadí), nikoli jedna konkrétní kategorie.

## 6.2 Rozhodovací strom

Rozhodovací strom [3], anglicky *decision tree*, je jedním ze základních nástrojů klasifikace. Podobá se vývojovému diagramu, ve kterém jednotlivé uzly testují některý z atributů. Každý test je definován klasifikačním pravidlem. Pro vybrání ideálního atributu k testování tak, aby se maximálně od sebe objekty odlišily, se využívá entropie. Na výstupu má standardně algoritmus na výběr ze 2 pokračujících větví. Podle výsledku testu algoritmus postupuje k dalšímu uzlu, než narazí na koncový uzel definující výslednou třídu. Tato hodnota se nazývá predikcí.

Mezi hlavní výhody patří jednoduchost a pochopitelnost, proces rozhodovacího algoritmu je možné jednoduše interpretovat. Zásadní nevýhodou je sklon k přeučení, který je možné také nazvat jako *overfitting*. To se nicméně podařilo vyřešit navázaným algoritmem, který se nazývá náhodný les (6.3.1). Dále jsou rozhodovací stromy považovány za nestabilní, protože i velmi malá změna v trénovacích datech může vést k zásadní změně struktury výsledného rozhodovacího stromu.

Vizualizaci rozhodovacího stromu na Iris datasetu je možné vidět na obrázku 6.1.




Obrázek 6.1: Rozhodovací strom [4]

## 6.3 Ensemble metody

Ensemble [3] je technika, při které se kombinuje více modelů tak, aby výsledný model měl lepší výkonnost než jednotlivé modely samostatně. Využívá se zejména u rozhodovacích stromů, kde se modely liší pouze ve zvolených trénovacích datech nebo způsobu, jakým byly vytvořeny. Ensemble pak díky těmto modelům dokáže zajistit robustnost vůči šumu, větší přesnost predikce a zlepšení schopnosti generalizace.

Existují různé typy ensemble metod, z nichž nejčastěji používané jsou:

**Averaging** – Při této metodě se predikce více modelů zprůměrují, aby se získaly výsledné hodnoty.

**Weighted averaging** – Funguje podobně jako průměrování, ale u této techniky se předpověď každého modelu váží podle jeho výkonu na validační množině. Modelům s lepším výkonem je přiřazena větší váha.

**Stacking** – Při této metodě trénujeme více modelů na stejné množině dat a jejich předpovědi pak použijeme jako vstup do nového modelu, který učíme předpovídat konečný výstup. Cílem je naučit model kombinovat silné stránky jednotlivých modelů.

**Bagging** – Další možností je trénování na různých podmnožinách trénovacích dat a kombinovat jejich predikce. To má za cíl snížit rozptyl predikcí a zlepšit výkonnost konečného modelu.

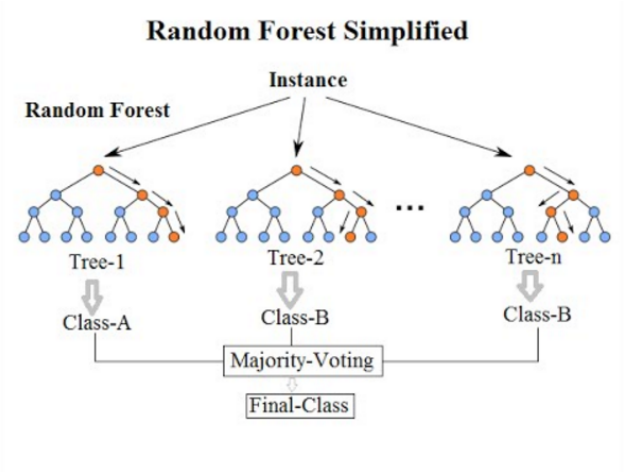
**Boosting** – Tato metoda je založená na trénování více slabších modelů, kdy každý následující model se zaměřuje na příklady, které předchozí modely vyhodnotily špatně. Konečná předpověď je pak váženým průměrem predikcí všech modelů.

V následující podsekcích popisuji náhodný les (kapitola 6.3.1) jako typického zástupce bagging metody a dále za boosting techniky zmiňuji gradientní boosting (kapitola 6.3.2), LightGBM (kapitola 6.3.3) a XGBoost (kapitola 6.3.4). Existující ale i další populární metody jako například AdaBoost a CatBoost.

### 6.3.1 Random forest

Náhodný les [3], anglicky *random forest* (RF), je skupinou velkého množství rozhodovacích stromů, zmíněných v předchozí sekci (6.2). Při klasifikaci je výstupem třída vybraná převažující většinou stromů. Náhodné lesy jsou obecně přesnější než rozhodovací stromy, protože nejsou náchylné na přeučení. Nevýhodou je větší výpočetní náročnost, neboť je pracováno s velkým množstvím rozhodovacích stromů.

Proces klasifikace přehledně ilustruje diagram 6.2.




Obrázek 6.2: Náhodný les [5]

### 6.3.2 Gradient boosting

Gradient boosting (GBM) [3] metoda iterativně vytváří množství rozhodovacích stromů, kdy každý následující se snaží opravit chyby předchozího stromu. V každé iteraci je natrénován nový rozhodovací strom tak, aby odpovídal zápornému gradientu ztrátové funkce vzhledem k predikcím předchozího stromu. Výsledná predikce je sumou predikcí jednotlivých stromů.

### 6.3.3 LightGBM

LightGBM [6] je postaven na podobných principech jako ostatní gradient boosting frameworky, ale přináší několik inovací v algoritmu učení. Zásadní změna spočívá ve využití techniky založené na histogramech, která umožňuje rychlejší a efektivnější trénování modelů. Mezi další výhody patří efektivní správa paměti a umožnění paralelního zpracování na více jádrech, případně ve výpočetním clusteru.

### 6.3.4 XGBoost

XGBoost (Extreme Gradient Boosting) [7] vylepšuje základní princip gradient boostingu tím, že používá regularizaci pro omezování přetrénování, optimalizuje váhy a využívá distribuovaného zpracování pro zrychlení trénování. Kromě toho XGBoost umožňuje přizpůsobení různých ztrátových funkcí a nastavení vlastních kritérií pro optimalizaci modelu.

Celkově se XGBoost stal oblíbenou volbou pro úlohy strojového učení díky vysoké přesnosti, krátké době trénování a škálovatelnosti. Využívá se v mnoha aplikacích, včetně doporučovacích systémů, klasifikace obrázků a zpracování přirozeného jazyka.

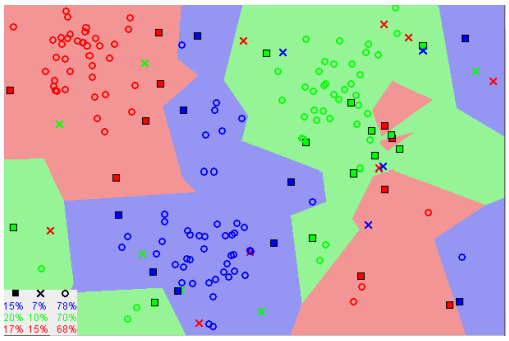
## 6.4 Naive Bayes

Naivní Bayesovský klasifikátor [3] patří do skupiny jednoduchých pravděpodobnostních klasifikátorů, které jsou založeny na Bayesově větě. Ta říká, že pravděpodobnost podmíněná na pozorování (např. výskytu určitého slova v textu) lze vypočítat jako součin pravděpodobností tohoto jevu a pravděpodobnost podmíněnou na pozorování vzhledem k jinému jevu.

## 6.5 Algoritmus k-nejbližších sousedů

Algoritmus k-nejbližších sousedů [3], anglicky *k-nearest neighbors* (KNN) je jednoduchý algoritmus pro strojové učení s učitelem. Základním předpokladem algoritmu je, že podobné entity se vyskytují v těsné blízkosti. Na základě výpočtu vzdálenosti mezi dvěma body v grafu můžeme určit jejich podobnost.

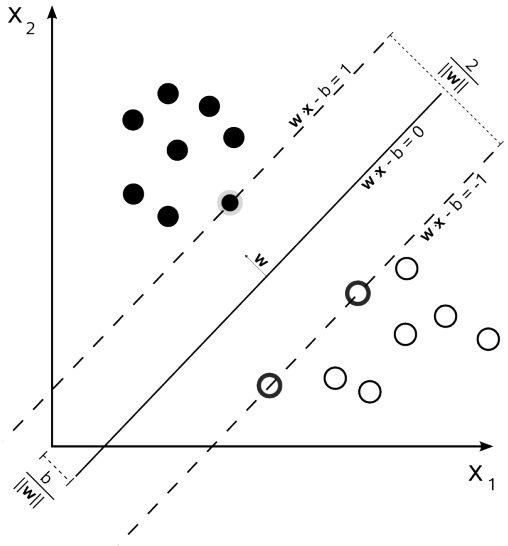
Mezi hlavní výhody patří jednoduchost implementace. Zároveň v této metodě není nutné ladit množství parametrů. Algoritmus je možný využít ke klasifikaci, regresi a vyhledávání podobných entit. Nevýhodou např. je, že s rostoucím objemem dat se algoritmus výrazně zpomaluje.




Obrázek 6.3: Algoritmus k-nejbližších sousedů [8]

## 6.6 Support vector machines

Metoda podpůrných vektorů [3] neboli *support vector machines* (SVM) je jedna z metod strojového učení s učitelem. Základem je lineární klasifikátor a trénovací data rozdělená do dvou tříd. SVM mapuje trénovací data na body v prostoru tak, aby se maximalizovala šířka mezery mezi těmito kategoriemi. Na popis pak stačí pouze body ležící na okraji, které nazýváme jako podpůrné vektory.




Obrázek 6.4: Support vector machines [9]

## 6.7 Neuronové sítě

Pro pochopení smyslu využití neuronových sítí, anglicky *artificial neural networks* (ANN), je zapotřebí si vysvětlit základní pojmy, princip fungování umělého neuronu a způsob jeho zapojení do sítě.

Tabulka 6.1: Aktivační funkce

| Funkce | Matematické vyjádření |
|---|---|
| Sigmoida | $f(x) = \frac{1}{1+e^{-x}}$ |
| Hyperbolický tangens | $f(x) = tanh(x)$ |
| Gaussova funkce | $f(x) = e^{-x^2}$ |
| Jednotkový skok (Heavisideova funkce) | $f(x) = \begin{cases} 0, & x < 0, \\ 1, & x \geq 0, \end{cases}$ |
| ReLU | $f(x) = max(0, x)$ |
| Softmax | $f_i(x) = \frac{exp(x_i)}{\sum_j exp(x_j)}$ |

Umělý neuron je základním stavebním kamenem neuronové sítě. Byl vytvořen jako zjednodušený model biologického neuronu. Jeho základním úkolem je vytvářet výstupní signál na základě vstupního potenciálu, který je určen sumou ohodnocených příchozích signálů, aktivační funkcí a prahem. Neuron může mít $N$ vstupů a pouze jeden výstup. Každý vstup neuronu je ohodnocen vahou, která se na počátku nastaví na určitou hodnotu (např. náhodně nebo pomocí funkce) a postupně se mění v průběhu trénování.

K sumě ohodnocených vah je poté přičten bias a výsledek je transformován pomocí aktivační funkce. Bias umožňuje posunout aktivační funkci na x-ové ose. Mezi nejpoužívanější patří aktivační funkce zmíněné v tabulce 6.1, přičemž v této práci byly využity nejčastěji funkce ReLU a softmax.

Perceptron je nejjednodušším modelem dopředné neuronové sítě (anglicky *feedforward neural network*) a skládá se pouze z jednoho neuronu. Zásadním omezením je jeho použití pouze na množiny, které jsou lineárně separovatelné. Využívá se tedy jako lineární klasifikátor.

Jak již bylo zmíněno, samostatný neuron má velmi omezené možnosti využití. Pro složitější úlohy tedy bylo navrženo propojit více neuronů do vrstev pomocí synapsí. Vrstvy těchto sítí lze rozdělit do základních kategorií podle typu použité vrstvy:

- Vstupní vrstva je vždy první vrstvou starající se o příjem informací. Obvykle jsou aktivační funkcí vstupy normalizovány v mezích limitních hodnot aktivačních funkcí. Počet vstupních neuronů odpovídá dimenzi vstupního vektoru.
- Skryté vrstvy se vyskytují v počtu 1 až $N$ vrstev. Lze ale také uvažovat o neuronové síti bez skrytých vrstev.
- Výstupní vrstva vytvoří výstupní produkt neuronové sítě jako výsledek zpracování vstupních dat předešlými vrstvami. Počet neuronů ve výstupní vrstvě běžně odpovídá počtu klasifikovaných tříd.

Všechny vrstvy jsou tzv. *fully connected*, což znamená, že všechny neurony v určité vrstvě jsou propojeny se všemi neurony v předchozí i následující vrstvě.

Backpropagation neboli algoritmus zpětného šíření chyby je metodou učení neuronových sítí. Využívá se k trénování vícevrstvých neuronových sítí při učení s učitelem, tedy na dvojicích vstupních objektů a požadovaných výstupů. Kvalita neuronové sítě je popsána chybovou funkcí a cílem této metody gradientního sestupu je minimalizovat chybovou funkci. Při trénování dochází ke změně vah vstupů neuronů.

## 6.8 Konvoluční neuronová síť

Konvoluční neuronové sítě (CNN) jsou typem neuronových sítí navržených speciálně pro úlohy zpracování obrazu. Základem jsou konvoluční vrstvy, které umožňují efektivní extrakci vlastností ze vstupních obrazových dat.

Každá konvoluční vrstva se skládá ze sady filtrů, které jsou aplikovány na vstupní obraz pomocí operace konvoluce. Filtry se skládají z matice vah, které se postupně posouvají po obraze a vytváří jednu výstupní hodnotu. Konvoluční vrstva produkuje tzv. příznakové mapy, které zachycují různé vlastnosti vstupního obrazu jako jsou hrany, rohy, tvary, textury apod.

Konvoluční sítě se nejčastěji využívají pro klasifikací obrazových dat a rozpoznávání objektů v obraze. Lze je ale také využít i ke zpracování textových dat.

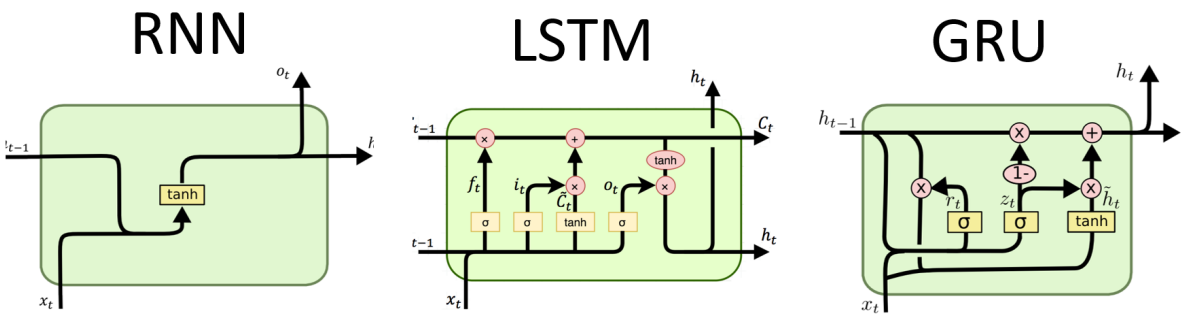
## 6.9 Rekurentní neuronová síť

Rekurentní neuronové sítě (RNN) jsou speciální architekturou neuronových sítí, která umožňuje pracovat se sekvencemi dat jako jsou například časové řady, textová data, zvukové záznamy apod. Jednou z nejvýznamnějších vlastností je jejich schopnost pracovat s proměnlivou délkou vstupní sekvence.

Tyto sítě se skládají z jedné nebo více rekurentních vrstev. Každá vrstva se skládá ze sady neuronů, které jsou vzájemně propojeny ve smyčce. Jednotlivé neurony přijímají na vstupu výstup z předchozích kroků zpracování vstupní sekvence a vstup v aktuálním časovém kroku. Dále vytváří výstup, který předávají do dalšího časového kroku.

Rekurentní sítě se nejčastěji využívají pro zpracování časově proměnných systémů, tedy např. zpracování přirozeného jazyka, predikce časových řad, řízení procesů, rozpoznávání řeči, detekce anomálií v síťovém provozu apod.

Existuje několik variant rekurentních sítí, z nichž mezi nejznámější patří LSTM (kapitola 6.9.1) a GRU (kapitola 6.9.2). Jednotlivé architektury obsahují další mechanismy, které jim umožňuje selektivně uchovávat nebo zapomínat informace z předchozích časových kroků, což pomáhá zlepšit schopnost zpracovávat dlouhodobé závislosti v datech.




Obrázek 6.5: Ukázka RNN (vlevo), LSTM (uprostřed) a GRU (vpravo) buňky [10]

### 6.9.1 LSTM

Long short-term memory (LSTM) je speciální druh zpětnovazebních neuronových sítí. Navrženy byly tak, aby překonaly tzn. *vanishing gradient* problém, který komplikuje učení dlouhodobých závislostí.

### 6.9.2 GRU

Gated recurrent unit (GRU) je dalším typem rekurentní neuronové sítě, která byla navržena pro zlepšení tradičních rekurentních sítí. Stejně jako u LSTM se i u GRU řeší problém se ztrátou informace v průběhu času a využívají se mechanismy pro zlepšení učení dlouhodobých závislostí. GRU buňky oproti LSTM obsahují méně parametrů, a tedy se rychleji trénují s potřebou menšího množství paměti.

## 6.10 Autoencoder

Autoenkodéry jsou neuronové sítě, které se běžně využívají v úlohách učení bez učitele, což zahrnuje například kompresi dat nebo odstranění šumu. Lze je však použít i v úlohách učení s učitelem, a to včetně úloh jako je klasifikace textu.

Skládají se ze dvou hlavních částí: *encoder* a *decoder*. Enkodér zpracovává vstupní data a vytváří z nich komprimované reprezentace. Dekodér pak přijímá tyto informace jako vstupy a snaží se rekonstruovat data zpět do původní dimenze.

V klasifikačních úlohách lze autoenkodéry využít k učení komprimovaných reprezentací textových vstupů, které lze poté předat klasifikátoru. Tento proces zahrnuje trénování autoenkodéru na souboru dat vstupních textů a následnou extrakci komprimovaných reprezentací. Ty lze poté využít jako vstupní vektor příznaků pro běžné klasifikátory.

Autoenkodéry zahrnují několik metod, přičemž mezi nejznámější patří variační autoenkodér (VAE), konvoluční autoenkodér (CAE) a rekurentní autoenkodér (RAE).

## 6.11 Transformer

Transformer modely jsou založeny na architektuře sítě, která se stala velmi populární pro zpracování přirozeného jazyka a strojový překlad. Tyto modely jsou schopné se učit kontextovou závislost a vztahy mezi slovy nebo tokem dat vstupujícím do sítě, což umožňuje dosáhnout lepší přesnosti.

Architektura se skládá ze dvou základních bloků, zvaných *encoder* a *decoder*. Kodér přijímá posloupnost vstupních tokenů a vytváří vektorovou reprezentaci celé posloupnosti o pevné délce. Dekodér převezme tuto reprezentaci a vygeneruje novou posloupnost tokenů.

Klíčovou inovací architektury transformeru je *attention mechanism* neboli mechanismus pozornosti. Ten umožňuje síti se zaměřit na různé části vstupní sekvence během generování výstupu. V mechanismu se každý token porovnává se všemi ostatními vstupními tokeny tak, aby se určila jeho relativní důležitost.

Mezi známé příklady modelů založených na architektuře transformer patří například BERT (kapitola 6.11.1), GPT (kapitola 6.11.2) a T5.

### 6.11.1 BERT

Bidirectional Encoder Representations from Transformers (BERT) je jeden z nejznámějších a nejúspěšnějších jazykových modelů založených na transformer architektuře. Byl vyvinut společností Google v roce 2018 a od té doby se stal jedním z nejpoužívanějších nástrojů pro řešení úloh v oblasti zpracování přirozeného jazyka.

Návrh spočívá v předtrénování z neanotovaného textu s využitím levého i pravého kontextu ve všech vrstvách. To znamená, že BERT dokáže zachytit kontext a význam slov ve větě na základě slov, která následují před a za ní.

Model využívá velkého množství trénovacích dat a natrénuje model tak, aby dokázal generovat vektorové reprezentace slov, vět a celých dokumentů. Tyto reprezentace pak mohou být využity pro různé úlohy jako například klasifikace, extrakce informací, překlad a další.

### 6.11.2 GPT

Generative Pre-trained Transformer (GPT) je skupina předtrénovaných modelů založených na architektuře transformer, vyvinutých společností OpenAI. Tyto modely jsou určeny pro zpracování přirozeného jazyka a umožňují generování textu, strojový překlad, odpovídání na otázky a další úlohy v této oblasti.

Model je předtrénován na velkém množství textových dat pomocí metody učení bez učitele, která se nazývá předtrénování. Během předtrénování se model učí předpovídat další slovo v sekvenci textu na základě předchozích slov. Tento krok umožňuje modelu naučit se obecné jazykové vzory a vztahy.

Mezi známé modely patří GPT-2, jehož předtrénování bylo provedeno s použitím rozsáhlé sady internetových dat. Celkem bylo využito více než 45 terabajtů textových dat. Model dokáže generovat

souvislé a kontextově relevantní věty. Umožnění generování realisticky vypadajícího textu vedlo k obavám ze zneužití ke generování falešných zpráv a vytváření *deepfake* obsahu.

Verze GPT-3 je jedním z nejmodernějších jazykových modelů. Jedná se o dosud největší a nejvýkonnější jazykový model s více než 175 miliardami parametrů. Dokáže generovat vysoce kvalitní souvislý text a byl použit v řadě aplikací, z nichž momentálně nejznámější je ChatGPT.

V nedávné době byla uvedena verze GPT-4, nicméně bohužel k tomuto modelu v tuto chvíli existuje veřejně jen velmi málo dostupných informací. Jedinou konkrétní informací, kterou se mi podařilo dohledat je přibližná cena trénování modelu, která se odhaduje v přepočtu na více než 2 miliardy českých korun.

# Kapitola 7

# Evaluace modelů

Evaluace modelů [3] strojového učení je nezbytný krok k určení jejich výkonnosti a výběru nejvhodnějšího modelu pro danou úlohu. Označuje proces, ve kterém se model testuje na nových, dosud neviděných datech, aby se ověřilo, jak dobře funguje v praxi. K vyhodnocení se využívá sada metrik v závislosti na dané úloze.

## 7.1 Křížová validace

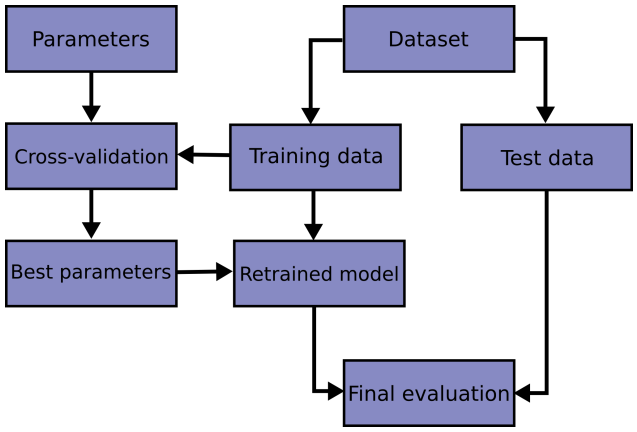
Křížová validace [3], anglicky *cross validation* je technika, která se využívá při procesu vyhodnocování kvality modelů. Jejím úkolem je rozdělení datové sady na několik různě velkých částí, které se zvlášť využívají k učení modelu a vyhodnocování výsledků.

### 7.1.1 Rozdělení na trénovací/validační/testovací sadu

Existuje několik typů křížové validace, z nichž nejběžnější zahrnuje rozdělení datové sady [3] na trénovací, validační a testovací množinu.

Trénovací data obvykle tvoří největší část datové sady a slouží k učení modelu, případně adaptaci vah v neuronové síti. Validační množina se využívá k ladění hyperparametrů a sledování výkonu během trénování modelu. Poslední část se nazývá testovací množinou a slouží k měření kvality predikcí výsledného modelu, zjištění, zda model dobře generalizuje na nová data a k celkovému vyhodnocení výkonu finálního modelu.

Postup vývoje modelu přehledně ilustruje diagram 7.1.




Obrázek 7.1: Grid Search Workflow [11]

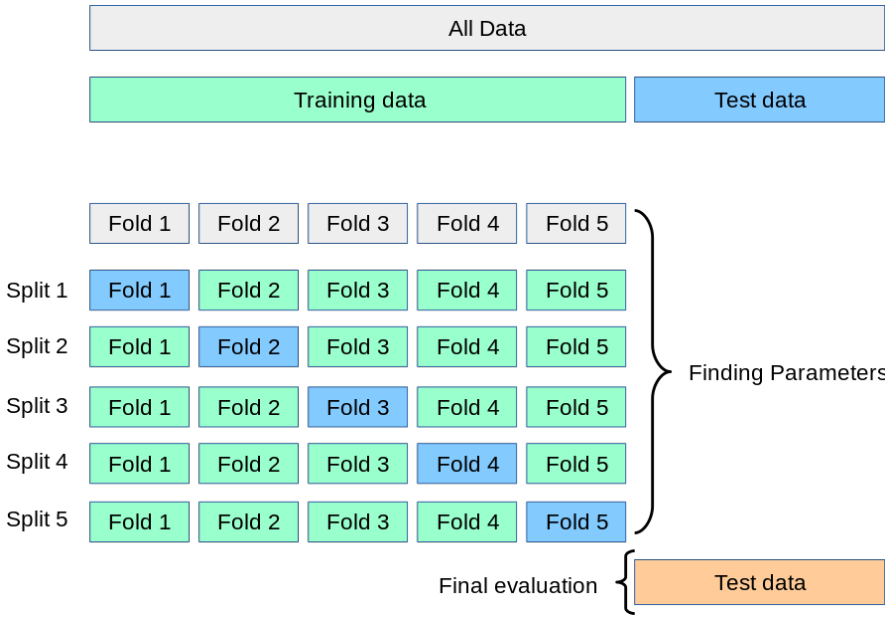
### 7.1.2 Rozdělení na trénovací/testovací sadu

Data lze také rozdělit i jen na trénovací a testovací množinu. V takovém případě se testovací sada využívá i k ladění hyperparametrů a může tímto lehce dojít k úniku dat testovací množiny. Důsledkem této chyby je, že hodnota výsledné metriky nemusí být kvalitní informací, jak dobře model funguje na dosud neviděných datech.

### 7.1.3 k-fold cross-validation

K-fold cross-validation [3] je technika zahrnující opakované rozdělení dostupných dat na $k$ stejně velkých podmnožin. Postupným opakováním křížové validace pomáháme snížit variabilitu odhadu výkonnosti modelu tím, že se postupně pro trénování, validaci i testování použijí všechna dostupná data. Zároveň je tato technika vhodná, pokud je dataset příliš malý a současně se zabraňuje *overfittingu*.

Proces opakovaného dělení datasetu přehledně ilustruje diagram 7.2.




Obrázek 7.2: Grid Search Cross Validation [12]

### 7.1.4 Stratified k-fold cross-validation

Techniku křížové validace lze dále rozšířit o stratifikaci [13], která zajišťuje, aby se v každé množině zachoval přibližně stejný poměr počtu vzorů jednotlivých tříd. Toho se využívá zejména pokud je datová sada nevyvážená.

## 7.2 Evaluační metriky

V závislosti na typu úlohy a charakteristikách datové sady existuje množství vhodných evaluačních metrik. Konkrétně pro klasifikaci jsou obecně považovány za vhodné metriky accuracy (kapitola 7.2.1), precision (kapitola 7.2.2), recall (kapitola 7.2.3) a F1 score (kapitola 7.2.4). Pro podrobnější vyhodnocení úspěšnosti klasifikace do jednotlivých tříd se využívá confusion matrix (kapitola 7.2.5).

Měření výsledků binárních klasifikačních úloh se popisuje 4 hodnotami:

**TP** – true positive (počet správně predikovaných pozitivních případů)

**FP** – false positive (počet špatně predikovaných pozitivních případů)

**FN** – false negative (počet špatně predikovaných negativních případů)

**TN** – true negative (počet správně predikovaných negativních případů)

V případě klasifikace do více tříd množství výstupních hodnot kvadraticky roste.

### 7.2.1 Accuracy

Přesnost, anglicky *accuracy*, je běžně využívanou metrikou pro klasifikační úlohy. Měří podíl správných predikcí oproti celkovému počtu vzorků.

Ačkoli může být užitečnou metrikou, není vždy tou nejlepší. Zejména pokud jsou třídy nevyvážené nebo se jedná o specifickou úlohu, kde je různý dopad falešně pozitivních a falešně negativních výsledků predikce, což nastává například v úloze detekce nemocí nebo poruch.

Matematický zápis je uveden v rovnici 7.1.

$$\text{accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}} \tag{7.1}$$

### 7.2.2 Precision

Preciznost, anglicky *precision*, je měření procenta správně předpovězených pozitivních vzorků oproti všem případům, které byly klasifikovány jako pozitivní. Jinými slovy měří, jak přesné jsou pozitivní predikce modelu. Matematické vyjádření je uvedeno v rovnici 7.2.

$$\text{precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{7.2}$$

### 7.2.3 Recall

Úplnost, anglicky *recall*, je zpětná vazba, která měří procento skutečně pozitivních případů, které model správně identifikoval. Matematický zápis je uveden v rovnici 7.3.

$$\text{recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{7.3}$$

### 7.2.4 F1 score

F1 skóre je objektivním měřítkem přesnosti modelu. Metrika se definuje jako harmonický průměr přesnosti a úplnosti. Matematický zápis je uveden v rovnici 7.4.

$$\text{F1 score} = \frac{2 \times \text{precision} \times \text{recall}}{\text{precision} + \text{recall}} \tag{7.4}$$

### 7.2.5 Confusion matrix

Matice záměny, anglicky *confusion matrix*, se využívá k porovnání predikovaných tříd oproti skutečným hodnotám v souboru dat. Z hodnot v tabulce je poté možné vypočítat další metriky.

# Kapitola 8

# Provedené experimenty

Jako nejdůležitější část této práce lze považovat vlastní experimentování s uvedenými technikami a modely.

Zdrojem dat pro všechny experimenty byla předzpracovaná data WikiCFP, která obsahovala 56599 záznamů z původně stažených 94270 stránek. Zbylých 37671 záznamů neobsahovalo žádné kategorie nebo byly zařazeny do kategorie, která nebyla optimalizována. Každý záznam v datové sadě obsahuje právě jednu z 10 kategorických štítků.

Dataset byl rozdělen v poměru 80:20 na trénovací a testovací sadu. Z dostupných atributů byl využit titulek CFP, neboť je na rozdíl od popisku vždy vyplněn. V navazujících experimentech bylo odzkoušeno množství modelů a jejich architektur. Výběr byl proveden tak, aby pokryl postupně všechny oblasti od lineárních modelů až po nejnovější modely s neuronovými sítěmi.

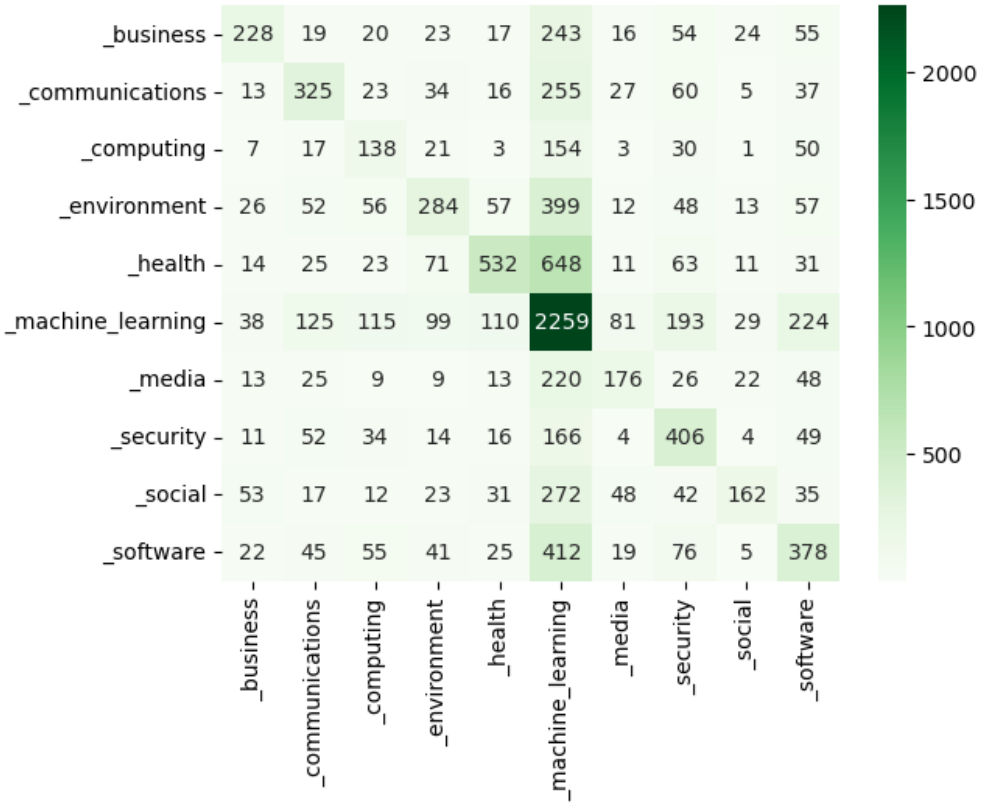
Z každého testovaného modelu byl vybrán zástupce s hyperparametry, při kterých měl model nejvyšší F1 skóre. To taky považuji za klíčovou metriku vzhledem k distribuci tříd v datové sadě. V tabulkách jsou dále uvedeny i ostatní metriky. Doba trénování a testování není u modelů uvedena, neboť to záviselo na dostupných kapacitách a jejich aktuálním využití. Změřené časy sloužily pouze k orientačnímu hodnocení. Detaily experimentů lze však dohledat v přiloženém souboru se zdrojovým kódem všech experimentů.

## 8.1 Doc2Vec + baseline modely

První experiment se týkal lineárních modelů, které byly zvoleny jako tzv. baseline. Oproti původnímu plánu se metoda rozšířila o další pokusy, neboť jejich implementace byla součástí stejné knihovny a používaly stejné API. Ve všech pokusech byla využita metoda Doc2Vec.

Tabulka 8.1: Výsledky experimentů s baseline modely

| | Trénovací data | Testovací data | | | |
|---|---|---|---|---|---|
| Název modelu | Accuracy | Accuracy | Precision | Recall | F1 score |
| Logistická regrese | 52,28 | 45,63 | 46,72 | 38,41 | 40,46 |
| Rozhodovací strom | 61,73 | 26,53 | 20,50 | 19,03 | 19,47 |
| Gaussian Naive Bayes | 38,99 | 38,29 | 41,38 | 31,38 | 31,38 |
| Bernoulli Naive Bayes | 41,08 | 36,62 | 36,23 | 24,58 | 25,73 |
| SVM | 68,78 | 50,72 | 64,45 | 38,02 | 44,26 |
| KNN | 100,00 | 52,96 | 48,86 | 48,64 | 48,71 |




Obrázek 8.1: Confusion matrix pro logistickou regresi

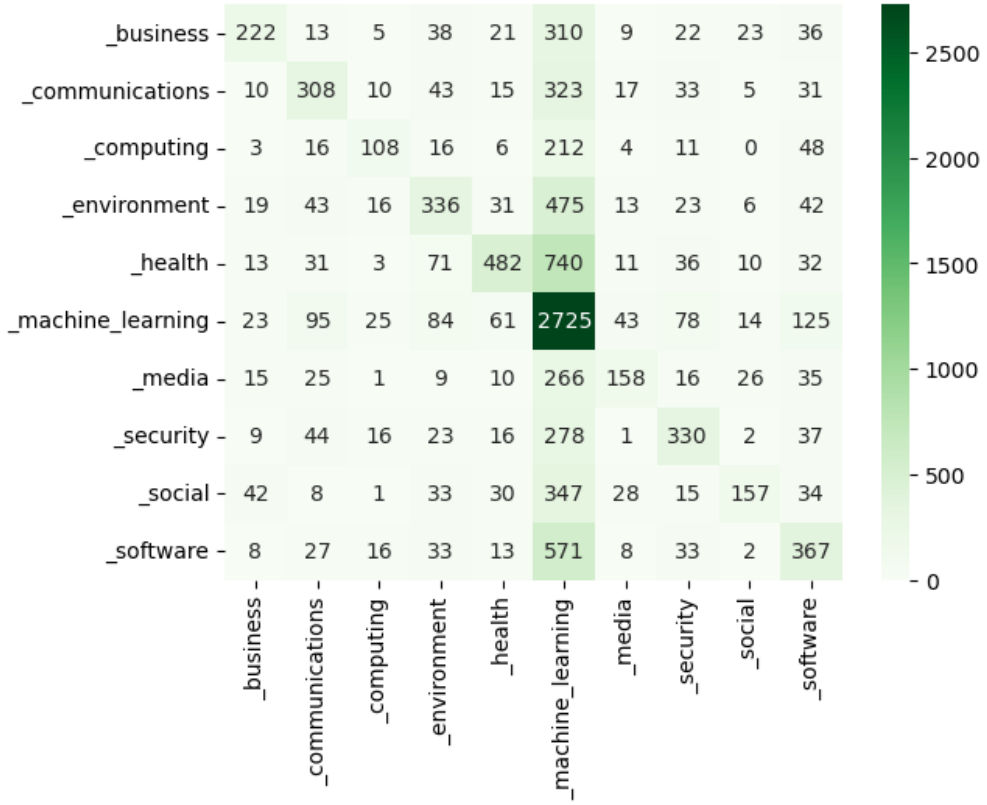
Z experimentů uvedených v tabulce 8.1 lze vyčíst, že nejlépe dopadl model KNN. Metoda také vynikala i v rychlosti trénování oproti SVM, která byla jako druhá a s mnohem delší dobou trénování.

## 8.2 Doc2Vec + ensemble techniky

Druhou kapitolou byly metody založené na spojení několika slabších modelů, tzv. ensemble. Jako první za kategorii bagging byl vybrán random forest následovaný několika boosting metodami.

Tabulka 8.2: Výsledky experimentů s ensemble technikami

| | Trénovací data | Testovací data | | | |
|---|---|---|---|---|---|
| Název modelu | Accuracy | Accuracy | Precision | Recall | F1 score |
| Random forest | 100,00 | 42,37 | 60,16 | 26,31 | 30,61 |
| AdaBoost | 47,88 | 37,65 | 36,08 | 30,26 | 30,84 |
| Gradient boosting | 62,85 | 44,67 | 46,22 | 34,78 | 37,76 |
| Histogram-based gradient boosting | 73,53 | 46,04 | 50,19 | 35,83 | 39,65 |
| XGBoost | 100,00 | 49,16 | 53,89 | 38,87 | 42,94 |




Obrázek 8.2: Confusion matrix pro XGBoost

Výsledky testovaných ensemble technik v tabulce 8.2 nejsou nijak překvapivé a odpovídají mému

očekávání. Zmíním zde akorát vyšší výpočetní náročnost. Zajímavostí XGBoost metody byla její efektivní paralelizace. Implementace dokázala natrénovat model během jednotek minut s využitím všech dostupných CPU vláken. Celkově tak spotřebovaný procesorový čas u jednoho experimentu se pohyboval i přes 48 hodin, což je vidět z anotací u spouštěných buněk v přiloženém souboru.

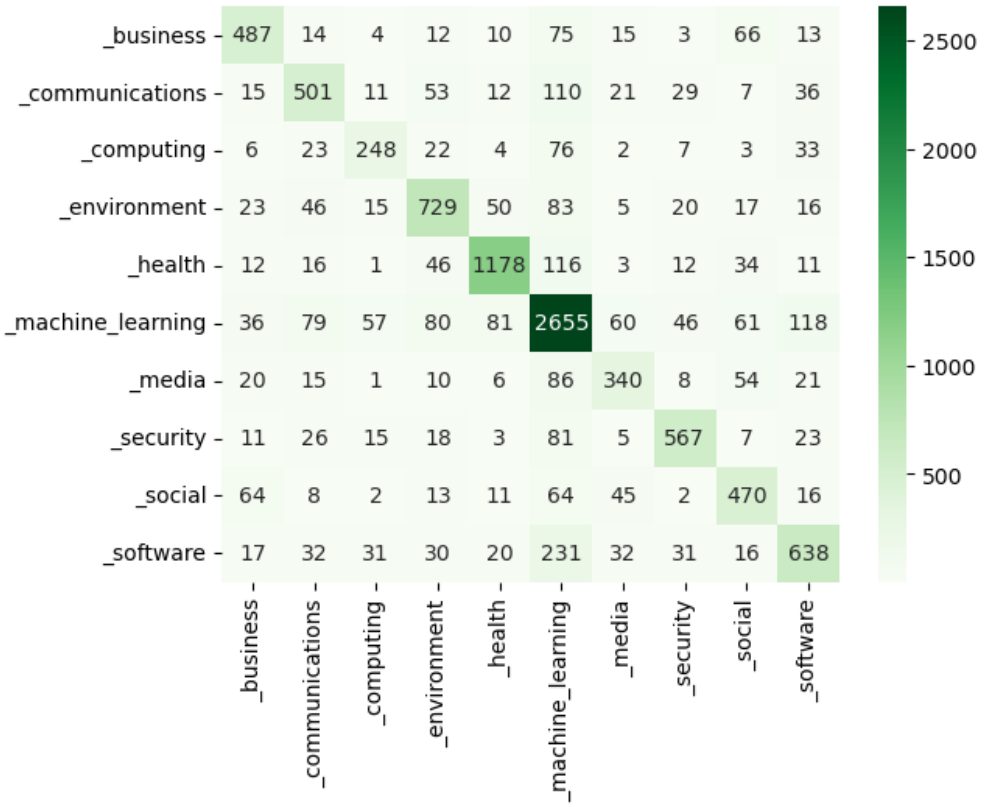
Z přiloženého obrázku 3 lze zřetelně vidět využití dostupných prostředků na serveru *argexp3*. Bez tohoto výpočetního stroje by experimenty nebylo možné provést.

## 8.3 Vlastní vektorizace + RNN

Na závěr byly vybrány neuronové sítě, a to konkrétně typ s rekurentními vrstvami. Využitý model byl založen na jedné LSTM vrstvě a jedné navazující GRU vrstvě. Neuronová síť byla využita s kombinací několika technik vektorových reprezentací textu.

Tabulka 8.3: Výsledky experimentů s rekurentními neuronovými sítěmi

| | Trénovací data | Testovací data | | | |
|---|---|---|---|---|---|
| Název modelu | Accuracy | Accuracy | Precision | Recall | F1 score |
| Vlastní vektorizace + RNN | 83,87 | 72,41 | 70,32 | 67,81 | 68,82 |
| GloVe + RNN | 72,84 | 70,97 | 68,12 | 67,36 | 67,43 |
| Doc2Vec + RNN | 60,79 | 49,81 | 50,85 | 41,68 | 43,80 |




Obrázek 8.3: Confusion matrix pro RNN model s vlastní vektorizací

Celkově model s vlastní vektorizační vrstvou vycházel velmi dobře při porovnání s předchozími experimenty. Obdobně s pozitivním výsledkem dopadl i stejný model s GloVe embeddingy. Problematickou kategorií se stala akorát skupina „machine learning“ často zaměňovaná se „software“. Ostatní kategorie nebyly špatně klasifikovány ve větším množství případů. Překvapivým zklamáním se stala celkově Doc2Vec vektorizace, se kterou se na základě výsledků experimentů nepodařilo vytvořit kvalitní embeddingy. To může být způsobeno nedostatkem trénovacích dat, špatnou volbou hyperparametrů, nevhodným předzpracováním dat způsobující šum v datech a nebo jejich kombinací.

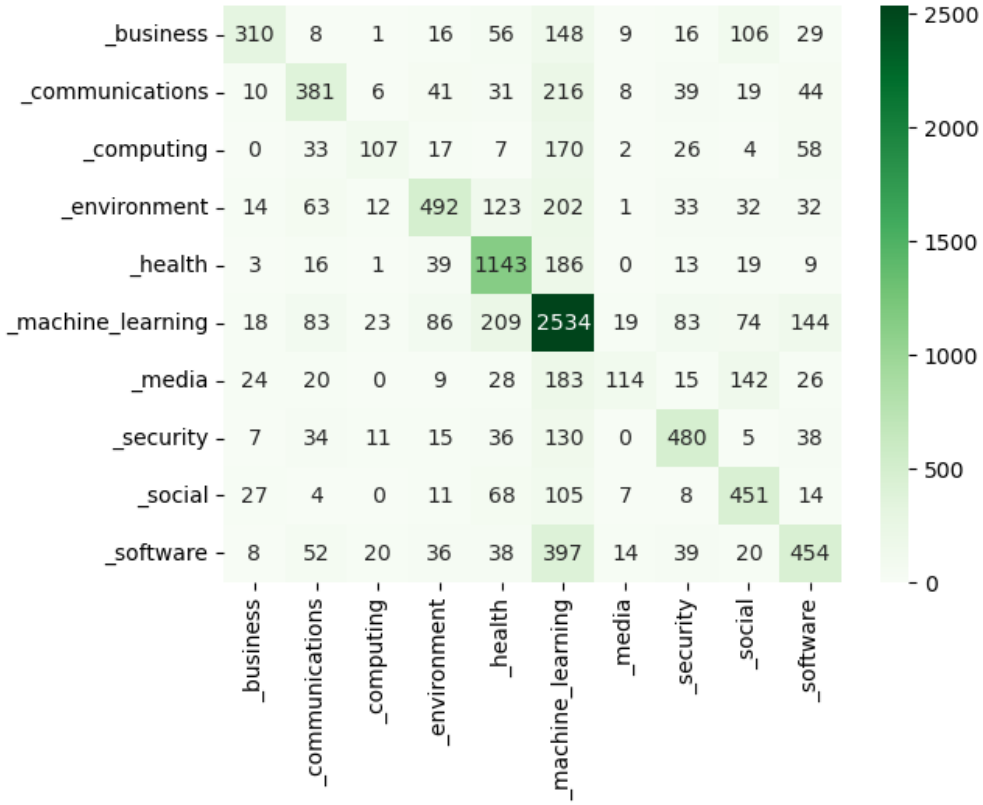
Z předchozích zkušeností z předmětu *deep learning* bylo očekáváné, že velká snaha o tuning hyperparametrů nepovede k zásadnímu zlepšení u tohoto typu sítě, a proto jsem spíše využíval empiricky zvolené hyperparametry v uvedených experimentech.

## 8.4 BERT (Hugging Face)

Za skupinu *state-of-the-art* modelů byl zvolen předtrénovaný transformer model BERT.

Tabulka 8.4: Experimentální výsledky

| | Trénovací data | Testovací data | | | |
|---|---|---|---|---|---|
| Název modelu | Accuracy | Accuracy | Precision | Recall | F1 score |
| BERT | 59,88 | 60,20 | 61,57 | 52,95 | 55,09 |




Obrázek 8.4: Confusion matrix pro BERT model

Výsledky jsem bohužel očekával u tohoto modelu mnohem lepší, nicméně zůstává zde prostor k mírnému zlepšení. Je zajímavou náhodou, že model měl na testovací množině mírně lepší hodnotu accuracy.

## 8.5 Shrnutí

V této části bych rád shrnul provedené experimenty. Pro testování modelů nebyla využita k-fold cross validation z důvodu větší časové náročnosti, která by vzhledem k povaze úlohy nepřinášela zásadně přesnější výstupy. V confusion matrices si můžeme všimnout podtržítka na začátku kategorie, to označuje pouze sloučené kategorie a zabraňuje tak záměně s původními kategoriemi.

Modely byly posuzovány zejména podle hodnoty F1 skóre, ale také podle času trénování a inference na testovací sadě, dále výpočetní náročnosti modelu, a to včetně možnosti akcelerace či paralelizace. Po zhodnocení všech aspektů se nejvýkonnějším modelem stal RNN s vlastní vektorizací, který dosáhl na F1 skóre s hodnotou 68,82 %.

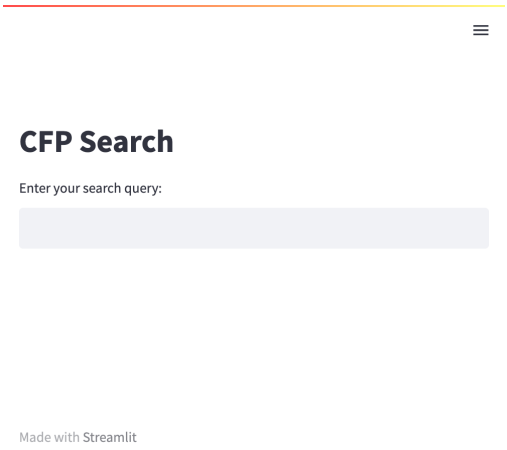
# Kapitola 9

# Webová aplikace

Webová aplikace byla realizována ve formě MVP sloužící účelu demonstrace nejlepšího modelu z kapitoly 8. Vybraným modelem, kterým se stal RNN s vlastní vektorizací, byla klasifikována CFP data a výsledky zaindexovány v Elasticsearch databázi.

Aplikace podporuje běžné fulltextové vyhledávání s podporou všech dostupných atributů. Pro doporučování relevantních konferencí bylo využito metody agregace výstupu fulltextového vyhledávání, kdy na základě klasifikovaných tříd byly vybrány podobné konference. Ty jsou dále filtrovány tak, aby se zobrazily pouze nadcházení konference seřazené od nejbližších dle data konání.

Výsledná aplikace byla vytvořena hlavně díky knihovny Streamlit popsané v kapitole 10.2.11. Funkční aplikaci jsem za účelem umožnění testování zveřejnil[1]. Pro uvedenou doménu byl také vygenerován Let's Encrypt certifikát. Celá aplikace je provozována uvnitř Docker kontejneru na virtuálním serveru, který je součástí virtualizační platformy VMware vSphere.




Obrázek 9.1: Uživatelské rozhraní

[1] https://demo.ing.marekhanus.cz/

# Kapitola 10

# Využité technologie

Na následujících stranách popisuji nejdůležitější technologie, které byly využity při zpracování této diplomové práce. Převážná část rešerše a vývoje aplikací byla realizována v programovacím jazyce Python s využitím množství knihoven pro zjednodušení práce a samostatných databází pro ukládání dat. U některých technologií také zhodnocuji i nalezená pozitiva a negativa.

Celou sadu nástrojů pro vytvoření aplikace, od programovacího jazyka, přes použité frameworky, databáze a další nástroje, lze také nazvat jako *tech stack*. Funkční a spolehlivé vývojové prostředí je velmi důležité, neboť zásadně ovlivňuje efektivitu práce.

## 10.1 Docker

Docker je populární platforma pro vytváření, přenášení a zajištění provozu spuštěných aplikací. Umožňuje uzavřít sestavenou aplikaci se všemi jejími závislostmi do tzv. kontejneru, který lze snadno sdílet a provozovat na libovolné platformě podporující technologii Docker. Všechny součásti mnou vytvořené aplikace jsou provozovány na této platformě a tímto umožňuji uživatelům velice jednoduše zprovoznit celou aplikaci s pomocí dostupných zdrojových kódů a předem vytvořených obrazů publikovaných na portálu Docker Hub[1].

### 10.1.1 Docker Compose

Součástí platformy Docker je sada nástrojů pro správu kontejnerů. Mezi ně patří i Docker Compose, který slouží ke správě aplikací složených z více kontejnerů. V souboru standardně nazvaném *docker-compose.yml* se definují jednotlivé služby, jejich uložiště a síťové propojení čímž se vytvoří výsledná aplikace. Umožňuje tímto administrátorům spravovat infrastrukturu ve zdrojovém kódu, anglicky se tento způsob nazývá *infrastructure as code*. Tímto lze spouštět celé aplikace s pomocí jediného příkazu na jakékoliv platformě. Typicky se Docker Compose využívá pro provoz závislých komponent

[1] https://hub.docker.com/

aplikace jako jsou relační či dokumentové databáze. Dále je takto možno provozovat aplikaci, která se skládá z více mikroslužeb, anglicky *microservices*.

## 10.2 Python

Python [14] je vysokoúrovňový interpretovaný programovací jazyk postavený na jednoduše čitelné syntaxi, která je vhodná i pro začátečníky. Nejčastěji slouží pro tvorbu webových aplikací, analýzu dat a vývoj umělých neuronových sítí. Lze jej také využít pro automatizační úlohy a v případě této práce pro automatické stahování a extrakci dat z webových stránek. Jako hlavní nevýhodu lze zmínit výkon, kde pro náročnější úlohy může být výrazně pomalejší při porovnání například s implementací v jazyce C++.

### 10.2.1 Project Jupyter

Jupyter [15] je užitečný softwarový projekt, jehož cílem je poskytnout uživateli webové prostředí pro interaktivní výpočty. Uživatel do zápisníků, nazvaných *notebooks*, může zapisovat programovací kód, rovnice, vizualizace i popisný text. Části kódu je možné rozdělovat do buněk, které lze spouštět jednotlivě nebo v určeném pořadí. Umožňuje také integraci interaktivních widgetů pro zobrazení složitých a dynamických vizualizací. Obsah celého dokumentu lze přehledně strukturovat, uložit jako soubor ve formátu JSON a sdílet s dalšími uživateli. Pro tuto práci byl využit v kombinaci s programovacím jazykem Python, nicméně podporuje i další jazyky jako například Julia a R.

Jupyter Notebook je původní prostředí poskytující základní funkce vytváření a úpravy zápisníků ve webovém prohlížeči. JupyterLab je nová generace s vylepšeným uživatelským prostředím a pokročilejšími funkcemi. Jedná se o kompletní interaktivní vývojové prostředí (IDE). JupyterHub umožňuje více uživatelům pracovat na jednom fyzickém prostředí, přičemž každému uživateli se vytvoří samostatná instance.

Alternativou mohou být služby poskytované cloudovými platformami, mezi které patří Google Colaboratory[2] a Paperspace Gradient Notebooks[3]. Obě služby jsou oblíbené díky možnosti přístupu k bezplatným výpočetním zdrojům včetně akcelerátorů jako jsou GPU a TPU zařízení.

V úvodu zpracování této práce jsem preferoval Google Colaboratory, nicméně vzhledem k opakovanému překračování limitů využití bezplatných zdrojů jsem poté začal využívat vlastní instalaci Jupyter Notebook na sdíleném fakultním serveru. Podrobněji využití obou technologií popisuji v sekcích 11.1 a 11.2.

[2] https://colab.research.google.com/
[3] https://www.paperspace.com/gradient/notebooks

### 10.2.2 Scrapy

Scrapy [16] je open-source framework, který se využívá k získávání dat z webových stránek. Poskytuje jednoduchý způsob pro prohledávání webu, extrakci dat a jejich ukládání ve strukturované podobě. Snadno lze takto například stahovat zprávy, informace o produktech a jejich aktuální ceny, případně zmíněné Call for Papers z webových stránek WikiCFP [17].

Framework poskytuje sadu funkcí pro zjednodušení práce s jednotlivými webovými boty, nazývanými *web crawlers*, případně *spiders*. Mezi hlavní výhody patří podpora cookies, možnost použití proxy serverů a nastavení hodnoty *user agent* pro identifikaci webového klienta. Další funkcí je logování a v případě chyb jsou užitečné nástroje pro ladění webových botů.

Extrahovat jednotlivé prvky z webové stránky lze pomocí selektorů XPath a CSS, kterými lze vybrat konkrétní HTML element, případně atribut dle zadaného vzoru. V této práci se mi více osvědčily CSS selektory, a to hlavně z důvodu jednoduchosti zápisu a možnosti získání cesty z vývojářských nástrojů implementovaných v prohlížečích rodiny Google Chrome [18]. Výstup extrahovaných dat lze poté uložit v běžných formátech typu JSON, CSV, XML nebo v tomto případě do dokumentové databáze MongoDB, podrobněji popsané v sekci 10.3.

### 10.2.3 Beautiful Soup

Beautiful Soup [19] je knihovna, která se využívá pro účely web scrapingu. Umožňuje získat data ze souborů HTML a XML pomocí objektového modelu dokumentu nazvaného DOM. Poskytuje tímto jednoduchý způsob procházení stromové struktury dokumentu a vyhledávání konkrétních HTML tagů nebo atributů.

Běžně se Beautiful Soup využívá v kombinaci s knihovnou *requests* pro provádění HTTP požadavků a *lxml* pro zpracování XML dokumentů.

Při provádění extrakce dat z WikiCFP [17] jsem postupně do této knihovny přepsal skripty původně napsané ve frameworku Scrapy z důvodu zjednodušení korekcí chyb. Při zjištění chyby v některém XPath nebo CSS pravidle bylo původně nutné vytěžit přes 150 tisíc URL adres znovu. Proto byly HTML výstupy z knihovny *requests* v první fázi uloženy do dokumentové databáze a až poté jako následující se pokračovalo zpracováním celé datové sady. Tímto se významně urychlilo získání kompletních strukturovaných informací, a tak jsem od knihovny Scrapy úplně upustil.

### 10.2.4 Pandas

Pandas [20] je open-source knihovna pro manipulaci s daty a k jejich základní analýze. Poskytuje nástroje pro předzpracování dat jako je slučování, filtrování, seskupování podle zadaného klíče a pivotaci. Užitečné jsou také funkce pro imputaci chybějících dat nebo odstranění celých záznamů s chybějícími daty.

Skládá se z datových struktur Series pro jednorozměrná data a DataFrame pro dvourozměrná data. Umožňuje také pracovat s časovými řadami. Využívá se proto jako univerzální nástroj pro analýzu dat a strojové učení.

Při propojení s knihovnami Matplotlib nebo Seaborn lze jednoduše vizualizovat data do grafů a diagramů. Za zmínku stojí jako alternativa knihovna Polars [21], která obsahuje optimalizované operace na více vláknech, podporu SIMD instrukcí a GPU akcelerátorů, čímž umožňuje efektivní práci s rozsáhlými datasety.

### 10.2.5 Matplotlib, Seaborn

Matplotlib [22] je populární knihovna pro vizualizaci dat poskytující širokou škálu nástrojů pro vykreslování grafů včetně možností přizpůsobení vzhledu, popisků os a legend grafu. Na tomto základu je postavena i knihovna Seaborn [23] poskytující vysokoúrovňové rozhraní pro vytváření přehledných grafů. Celkově lze říct, že Seaborn je bohatá a flexibilní knihovna pro tvorbu estetické grafiky. Vhodným datovým vstupem jsou informace uložené v datových strukturách Pandas.

Jako alternativu lze zmínit knihovnu Plotly [24] implementovanou do mnoha programovacích jazyků.

### 10.2.6 NLTK

Natural Language Toolkit [25] je knihovna poskytující prostředky pro práci s jazykovými daty. Nabízí přístup k jazykovým korpusům a algoritmům pro zpracování a analýzu dat přirozeného jazyka.

Jednou z funkcí je tokenizace pro rozdělování textu na slova nebo fráze. Punkt Sentence Tokenizer slouží k tokenizaci textu do vět. Modul obsahuje sadu pravidel pro identifikaci interpunkčních znamének na konci věty a také sofistikovanější metody pro detekci zkratek a jiných speciálních případů.

Užitečná je také funkce FreqDist, kterou lze využít k vypočtení frekvence výskytu slov nebo frází v korpusu. Výstup lze nazvat také jako četnost slov.

Dále NLTK obsahuje seznamy stop slov pro několik jazyků. Tato slova se v přirozeném jazyce běžně využívají, ale obvykle nijak nemění význam vět. Během předzpracování textových dat většinou dochází k jejich odstranění, aby se zvýšila přesnost jazykových modelů využitých například ke klasifikaci textu nebo analýze sentimentu. Podrobněji postup odstranění stop slov popisuji v sekci 3.1.2.

Důležitou funkcí je také stematizace a lemmatizace. Knihovna obsahuje řadu metod, mezi které patří Lancaster Stemmer, Porter Stemmer a WordNet Lemmatizer. Využitým zástupcem této řady byl nakonec vybrán lemmatizér, konkrétně WordNet Lemmatizer. Více o stematizaci a lemmatizaci popisuji v sekcích 3.1.3 a 3.1.4.

Mezi užitečné funkce patří také rozpoznávání slovního druhu (POS), pojmenovaných entit (NER), analýza sentimentu a mnoho dalších.

### 10.2.7 scikit-learn

Scikit-learn [13] je open-source knihovna pro strojové učení, která poskytuje řadu algoritmů pro klasifikaci, regresi, shlukování a redukci dimenzionality. Obsahuje také nástroje pro předzpracování dat, výběr modelu a jeho vyhodnocení.

Základními funkcemi jsou algoritmy pro strojové učení, mezi které lze zařadit lineární modely (kapitola 6.1), modely založené na rozhodovacích stromech (kapitola 6.2), Naive Bayes (kapitola 6.4), KNN (kapitola 6.5), SVM (kapitola 6.6) a další. Mezi pomocné funkce patří předzpracování, normalizace dat, rozdělení datové sady, křížová validace (kapitola 7.1) a metody pro měření přesnosti modelu. Mezi tyto metriky patří například accuracy (kapitola 7.2.1), precision (kapitola 7.2.2), recall (kapitola 7.2.3), F1 score (kapitola 7.2.4), ROC křivky a confusion matrix (kapitola 7.2.5).

### 10.2.8 Gensim

Gensim [26] je známou knihovnou pro zpracování přirozeného jazyka. Knihovnu vytvořil v roce 2008 český výzkumník Radim Řehůřek při práci na své disertační tézi [27], ve které se zabýval detekcí plagiátů. Mezi běžné užití patří analýza podobnosti dokumentů, klasifikace textu a *topic modeling*. Zajímavou součástí je implementace Doc2Vec modelu pro vytváření vektorové reprezentace dokumentů, který podrobněji popisuji v sekci 4.2.2. Výhodou je efektivní správa paměti i při zpracování velkých textových kolekcí, implementace evaluačních metrik a podpora množství populárních algoritmů včetně Word2Vec modelu, který detailněji popisuji v sekci 4.1.3.

### 10.2.9 TensorFlow

Mezi klíčové technologie této práce lze zařadit framework TensorFlow [28]. Využívá se pro úlohy strojového učení. Pochází z laboratoře výzkumného týmu Google Brain a v roce 2015 byl i včetně zdrojového kódu uvolněn veřejnosti.

Umožňuje vytváření a trénování modelů strojového učení pomocí nízkoúrovňového programovacího rozhraní, čímž umožňuje pokročilejším uživatelům vysokou flexibilitu a kontrolu nad vytvářeným modelem. Obsahuje také sadu funkcí pro předzpracování dat, evaluaci modelů a nástroj TensorBoard pro vizualizaci metrik. Další výhodou je podpora GPU a TPU akcelerátorů a dále distribuovaných výpočtů na více výpočetních zařízeních.

Knihovna Keras [29] poskytuje vysokoúrovňové rozhraní API postavené nad TensorFlow. Účelem tohoto nástroje bylo umožnit uživatelům za pomoci intuitivního rozhraní rychlé vytváření, trénování a experimentování s DL modely. Původně byl vyvinut jako samostatná knihovna, ale později byl do TensorFlow integrován. Podporuje širokou škálu architektur neuronových sítí, a to včetně CNN, RNN a architektury transformer.

Za alternativu lze považovat populární knihovnu PyTorch [30], která poskytuje obdobně uživatelsky přívětivé API rozhraní a flexibilní nástroje pro práci s DL modely. Obě knihovny jsem za dobu studia využíval. Nad výběrem ideální z nich se vedou velké debaty, nicméně mé rozhodnutí nakonec padlo právě pro *TensorFlow-based* prostředí, a to z důvodu největších zkušeností s vývojem RNN na této platformě.

### 10.2.10 Hugging Face

Hugging Face 🤗 [31] je společnost zaměřující se na vývoj nástrojů pro zpracování přirozeného jazyka. Přispívají k vývoji nejmodernějších modelů hlubokého učení a stojí za vytvořením populární knihovny Transformers.

Transformers [32] je open-source knihovnou poskytující předtrénované modely a umožňuje jejich jednoduchou integraci do stávajících projektů. Nejčastěji se využívá pro analýzu sentimentu, klasifikaci textu, strojový překlad a *question answering* neboli zodpovídání otázek. Populárními modely z architektury transformers jsou například BERT, GPT-2 včetně množství jejich derivátů jako jsou DistilBERT, RoBERTa a mnoho dalších. Tyto modely jsou ideální pro *transfer-learning*, a proto byly některé využity i v této práci.

Za zmínku stojí i další nástroje pro NLP, mezi které patří knihovny Tokenizers, Datasets, cloudová platforma Hugging Face Hub a vytvořený blog[4].

### 10.2.11 Streamlit

Streamlit [33] je open-source knihovna pro Python, která slouží k vytváření interaktivních webových aplikací. Samotná knihovna se stará o vykreslování webových stránek a umožňuje uživateli se soustředit na logiku fungování aplikace. Vzhledem ke stručné a intuitivní syntaxi je možné napsat celou aplikaci na pár řádcích kódu.

Nejčastěji se Streamlit využívá při analýze dat nebo k vytváření webového rozhraní nad DL modely. Pro svou jednoduchost si jej oblíbili datoví výzkumníci, kterým pomocí jednoduchého rozhraní umožňuje vytvářet vizualizace dat včetně zadávání uživatelských vstupů. Z těchto důvodů byla také využita k realizaci POC aplikace pro demonstraci vytvořených modelů.

## 10.3 MongoDB

MongoDB [34] je populární open-source databázový systém orientovaný na ukládání dokumentů. Řadí se mezi NoSQL databáze a umožňuje vysokou flexibilitu, tedy škálování a režim vysoké dostupnosti díky zabudovanému systému replikací. Data se ukládají do tzv. kolekcí ve formě binárních JSON (BSON) dokumentů, které mohou obsahovat kromě běžných typů také binární data, datum, čas i další datové typy.

[4] https://huggingface.co/blog

Tento databázový systém se často využívá při zpracování velkých objemů dat. Snadná je také integrace do aplikací v jazyce Python. V této práci je databáze využita jako úložiště všech surových dat HTML stránek z WikiCFP, dále předzpracovaných CFP v JSON strukturách a společných metadat. Takovéto využití lze pojmenovat jako *feature store*, tedy se jedná o centrální úložiště zpracovaných dat, která se využívají jako přímý vstup při trénování DL modelů.

## 10.4 Elasticsearch

Elasticsearch [35] je vyhledávací nástroj postavený na platformě Lucene. Poskytuje rychlé vyhledávání a možnosti škálování pro velké objemy dat včetně zajištění vysoké dostupnosti. Pomocí REST API zpřístupňuje rozhraní pro indexování a vyhledávání. Kromě základního fulltextového vyhledávání poskytuje i vyhledávání fasetové, geoprostorové a agregaci dat.

Obdobně i dříve zmíněné MongoDB v sekci 10.3 poskytuje možnosti indexování a vyhledávání, nicméně Elasticsearch byl pro vyhledávání navržen a poskytuje mnohem robustnější vyhledávací indexy. Podpora typů dotazů je zde mnohem rozšířenější jako například fuzzy vyhledávání. V této práci byl nástroj využit pro indexaci a vyhledávání dokumentů v demonstrační aplikaci.

### 10.4.1 Open Distro for Elasticsearch

Za zmínku také stojí nástroj Open Distro for Elasticsearch [36], který obsahuje implementaci k-NN algoritmu [37]. Díky tomu umožňuje aproximované vyhledávání na bázi vzdálenosti vektorů ve vysokodimenzionálních prostorech. Využívá se pro obrazové a textové vyhledávání, doporučovací systémy a k detekci anomálií.

Vstupem algoritmu jsou vysokodimenzionální vektory, například vektorové reprezentace textů generované DL modely. Vektorové vyhledávání díky již řešeným vektorovým reprezentacím v kapitole 4 by mohlo být zajímavým rozšířením této práce. Vyhledávání pomocí vektorů do svého fulltextového vyhledávání již implementovala například společnost Google [38] a také český Seznam.cz [39].

# Kapitola 11

# Výpočetní hardware

Trénování DL modelů často vyžaduje značné využití výpočetních kapacit. Již zmíněné nejpoužívanější frameworky TensorFlow a PyTorch v sekci 10.2.9 umožňují provádění matematických výpočtů na CPU, akcelerovaných GPU i specializovaných TPU.

Nejjednodušší řešení je využití široce dostupných CPU, nicméně v případě větších modelů může trénování trvat příliš dlouho, a to i v případě rozšířené podpory vektorových instrukcí. V případě GPU akcelerátorů se nejčastěji používají grafické karty značky NVIDIA s využitím CUDA technologie, která poskytuje násobné zrychlení díky velkému množství CUDA jader.

Zajímavé možnosti nabízí speciální TPU [40] optimalizované pro trénování neuronových sítí. Zařízení TPU nabízí cloudové řešení od Google a také je možné jej vyzkoušet na platformě Google Colaboratory.

## 11.1 Google Colaboratory

Google Colaboratory[1], zkráceně také Google Colab, je bezplatná cloudová služba od společnosti Google, která poskytuje uživatelům přístup k prostředí založeném na platformě Jupyter. Umožňuje připojení ke GPU a TPU instancím, a proto se stala oblíbeným nástrojem pro strojové učení a datovou analýzu. Poskytuje předinstalované balíčky NumPy, Pandas, Matplotlib, scikit-learn, TensorFlow, PyTorch a mnoho dalších.

Zásadní výhoda spočívá v jednoduchosti použití ve webovém prohlížeči bez nutnosti instalovat jakýkoliv další software. Mezi nevýhody patří omezení přístupu k bezplatným instancím a limity dostupných kapacit pamětí. V placených verzích Google nabízí výkonnější instance, nicméně i tak jsou parametry dle mého názoru dost omezené například v neumožnění dalšího navýšení paměti a nedostupnosti prostředí s více GPU.

[1] https://colab.research.google.com/

Existuje množství obdobných služeb, z nichž nejpopulárnějšími jsou Paperspace Gradient Notebooks[2], Kaggle Code[3] a Databricks Notebooks[4].

## 11.2 Výpočetní server na FEI (argexpr3)

V závěru zpracování této práce jsem dostal zajímavou příležitost, a to přístup k výpočetnímu serveru *argexpr3* umístěnému v serverovně katedry informatiky. Jedná se o úžasný kousek hardware s extrémními kapacitami určený primárně pro bioinformatické výpočty. Jedná se o typ HPE Superdome Flex Server [41] v konfiguraci s 20 procesory Intel Xeon Gold 6154 s taktem 3,7 GHz; 120 moduly operačních pamětí DDR4 s kapacitou 64 GB a 2 grafickými kartami NVIDIA Tesla V100 PCIe 32GB. Nicméně uvedené kapacity nejsou maximem, které tato platforma od HPE podporuje.

Navýšení výpočetních kapacit umožnilo testování velkých modelů založených na transformer architektuře. Tyto modely nebylo možné načíst na bezplatných službách jako je Google Colaboratory, případně jejich použití bylo velmi komplikované z důvodu omezených kapacit pamětí. Navíc v poslední době postupně narůstá množství dostupných modelů vyžadující více VRAM paměti, než kolik obsahují nejnovější grafické karty dostupné pro běžné spotřebitele.

Hlavní výhodou vlastních výpočetních zdrojů oproti cloudovým službám je jejich nepřetržitá dostupnost, což umožnilo trénovat náročné modely bez přerušení i několik hodin. S tím souvisely i nadlimitní kapacity pro využití jakéhokoliv modelu a množství dat. Další výhodou byla plná kontrola nad prostředím, včetně volby balíčků, verzí a jejich konfigurace. Mezi nevýhody lze zařadit nutnost technické zdatnosti pro zprovoznění výpočetního prostředí, řešení problémů s nekompatibilními balíčky, a hlavně časovou náročnost potřebnou pro vytvoření takového prostředí oproti připraveným řešením v cloudových službách.

[2] https://www.paperspace.com/gradient/notebooks
[3] https://www.kaggle.com/code
[4] https://www.databricks.com/product/collaborative-notebooks

Tabulka 11.1: Porovnání dostupných výpočetních kapacit

| Parametr | Google Colaboratory[1] | | | HPE Superdome Flex Server | |
|---|---|---|---|---|---|
| | pouze CPU | GPU | TPU | argexpr3 server | maximální konfigurace |
| CPU model | Intel Xeon[2] | Intel Xeon[2] | Intel Xeon[2] | Intel Xeon Gold 6154 | Intel Xeon Platinum[3] |
| CPU počet | 1 | 1 | 1 | 20 | 32 |
| CPU jádra | 1 | 1 | 1 | 360 | 896 |
| CPU vlákna | 2 | 2 | 2 | 720 | 1792 |
| RAM paměť | 12,7 GB | 12,7 GB | 12,7 GB | 7,2 TB | 48 TB |
| Disková kapacita | 107,7 GB | 78,2 GB | 107,7 GB | 129 TB[4] | 1+ PB[5] |
| GPU model | – | NVIDIA Tesla T4 | – | NVIDIA Tesla V100 PCIe 32GB | NVIDIA A100 80GB PCIe |
| GPU počet | – | 1 | – | 2 | 8 |
| GPU CUDA jádra | – | 2560 | – | 2x 5120 | 8x 6912 |
| GPU Tensor jádra | – | 320 | – | 2x 640 | 8x 432 |
| GPU paměť[6] | – | 16 GB GDDR6 | – | 2x 32 GB HBM2 | 8x 80 GB HBM2e |
| TPU model | – | – | TPU v2 | – | –[7] |
| TPU počet | – | – | 1 | – | – |
| TPU jádra | – | – | 8 | – | – |
| TPU paměť | – | – | 16 GiB HBM | – | – |

[1] bezplatné instance
[2] virtualizováno, různé modely
[3] modely 3. generace
[4] součet kapacit hlavních datových úložišť (29+29+71 TB) bez prostoru pro zálohování
[5] pouze teoretický součet za použití NVMe a SFF/LFF disků, může se lišit v závislosti na RAID konfiguraci
[6] VRAM
[7] není podporováno

# Kapitola 12

# Závěr

Cílem této práce bylo prozkoumání metod pro automatickou klasifikaci textu. Zpracování bylo komplexního charakteru, a to od zkoumání dostupných datových sad až po implementaci demonstrační aplikace.

V úvodu byla provedena rešerše datových zdrojů uvedených v zadání práce. Hned ze začátku při zpracování dat nastaly komplikace a postupně jsem zjistil, že téma získání kvalitních dat není triviálním problémem. Jako první jsem zjistil, že třídy v datech WikiCFP obsahují velký šum, neboť kategorie píšou přispěvatelé ručně do připravených textových polí. Následovala chybějící možnost stažení dat a nenalezení volně dostupné verze ke stažení. Dalším zklamáním byla neexistence tříd v datech DBWorld. Z tohoto důvodu jsem provedl rešerši množství dalších datových zdrojů, ve kterých jsem bohužel nenašel zdroj ekvivalentní databázi WikiCFP s kvalitně anotovanými třídami.

Pro získání dat z WikiCFP byly implementovány stahovací skripty. Samotný proces stažení trval poměrně dlouho, než skript dokázal zpracovat více než 155 tisíc stránek. Jednotlivé webové stránky musely být stahovány postupně po jedné, aby nedošlo k přetěžování databáze a porušení uvedených podmínek. Oproti tomu data DBWorld sice šlo jednoduše stáhnout pomocí RSS kanálů v mailing listu, nicméně vzhledem k neexistenci tříd nebylo možné provést evaluaci. V úvahu přicházela možnost ruční anotace dat tisíců příspěvků, ale to by bylo mimo rozsah této práce.

Po získání dat následovalo předzpracování. Příspěvky ve WikiCFP nemají jednotnou strukturu obsahu což se naštěstí později neukázalo jako komplikace, neboť se i tak některé modely podařilo naučit. Bohužel to nelze říct o kategoriích, kterých existují tisíce. Spousta z nich je nesmyslných, kategorie nemají jednotnou strukturu, objevují se duplicity a už vůbec nejsou disjunktní, jak se u klasifikace očekává. Pro pokračování v práci nezbyla jiná možnost než najít řešení. To po několika neúspěšných pokusech nakonec spočívalo v ručním napárování 300 nejčastějších a relevantních kategorií do 10 připravených skupin. Párování bylo časově velmi náročné, neboť jak jsem již v práci uvedl, jedině z kvalitních dat lze naučit dobrý model.

Práce pokračovala množstvím experimentů, které byly limitovány dostupností služby Google Colaboratory. Necelé dva měsíce před odevzdáním jsem naštěstí po konzultaci s vedoucím práce

získal možnost přístupu na výkonný výpočetní server katedry informatiky. Umožnění vyzkoušení výpočetně náročnějších modelů bylo pro mě také velkou motivací k dokončení této práce.

Součástí práce je také i jednoduchá demonstrační aplikace, která byla i díky dostupným nástrojům implementována velice rychle. Webová aplikace obsahuje funkce klasifikace a doporučování konferencí na základě klasifikované kategorie.

V posledních dvou kapitolách jsem věnoval poměrně hodně obsahu využitým knihovnám, nástrojům a hardware, na kterém byly prováděny výpočty. Použití nástrojů jsem se snažil odůvodnit i se zmíněním jejich výhod i některých nevýhod.

Celkově bylo zpracování poměrně zajímavé a rozhodně se nedá říct, že by práce byla v rozsahu stažení dostupného datasetu a vyzkoušení pár metod. Oproti původním záměrům se práce o dost rozšířila a obsah se vyvíjel postupně dle nastalých komplikací a zkoumaných metod.

Zajímavým rozšířením práce by mohlo být zavedení zmíněného vektorového namísto současného fulltextového vyhledávání. Další možností je zlepšení filtrace příspěvků pomocí polohy konference, neboť každá konference má textově uvedenou lokaci a pomocí techniky geocodingu by bylo možné převést informaci na GPS souřadnice, které by umožnily vyhledávat konference v okolí zadaného místa.

Práce by se mohla rozšířit také jiným směrem, například implementací *zero shot classification* metod nebo testováním komerčních GPT-4 modelů od organizace OpenAI.

Zajímavou otázkou k obhajobě by bylo objasnění důvodu, proč jsem nezkusil validovat modely nad některými uvedenými databázemi. To vycházelo primárně z důvodu nekompatibilních kategorií, případně nedostupných anotací tříd, jejichž absence by neumožňovala experimenty vyhodnotit.

Na závěr uvedu, že ač jsem po prvotním detailním prostudování zadání považoval zvolené téma za zásadní chybu, tak při psaní tohoto odstavce považuji téma za užitečné a cením si svého času, který jsem za 3 roky do zpracování této práce a studia obecně postupně investoval.

Obrázek 12.1: Sbohem a šáteček.

# Literatura


1. MANNING, C.D.; RAGHAVAN, P.; SCHÜTZE, H. *Introduction to Information Retrieval*. Cambridge University Press, 2008. ISBN 9781139472104. Dostupné také z: `https://books.google.cz/books?id=t1PoSh4uwVcC`.
2. WANG, Congcong; NULTY, Paul; LILLIS, David. A Comparative Study on Word Embeddings in Deep Learning for Text Classification. In: *Proceedings of the 4th International Conference on Natural Language Processing and Information Retrieval*. Seoul, Republic of Korea: Association for Computing Machinery, 2021, s. 37–46. NLPIR 2020. ISBN 9781450377607. Dostupné z DOI: `10.1145/3443279.3443304`.
3. HASTIE, Trevor; FRIEDMAN, Jerome; TIBSHIRANI, Robert. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. 1. vyd. New York: Springer New York, NY, 2013. ISBN 978-0-387-21606-5.
4. [Online] [cit. 2023-04-28]. Dostupné z: `https://scikit-learn.org/stable/_images/sphx_glr_plot_iris_dtc_002.png`.
5. [Online] [cit. 2023-04-28]. Dostupné z: `https://upload.wikimedia.org/wikipedia/commons/7/76/Random_forest_diagram_complete.png`.
6. *Welcome to LightGBM's documentation! — LightGBM 3.3.5.99 documentation* [online] [cit. 2023-04-23]. Dostupné z: `https://lightgbm.readthedocs.io/en/latest/`.
7. *XGBoost Documentation — xgboost 1.7.5 documentation* [online] [cit. 2023-04-23]. Dostupné z: `https://xgboost.readthedocs.io/en/stable/`.
8. [Online] [cit. 2023-04-28]. Dostupné z: `https://upload.wikimedia.org/wikipedia/commons/e/e9/Map1NNReducedDataSet.png`.
9. [Online] [cit. 2023-04-28]. Dostupné z: `https://upload.wikimedia.org/wikipedia/commons/2/2a/Svm_max_sep_hyperplane_with_margin.png`.
10. [Online] [cit. 2023-04-28]. Dostupné z: `http://dprogrammer.org/wp-content/uploads/2019/04/RNN-vs-LSTM-vs-GRU.png`.
11. *Grid Search Workflow* [online] [cit. 2023-04-22]. Dostupné z: `https://scikit-learn.org/stable/_images/grid_search_workflow.png`.

12. *Grid Search Cross Validation* [online] [cit. 2023-04-22]. Dostupné z: https://scikit-learn.org/stable/_images/grid_search_cross_validation.png.

13. PEDREGOSA, Fabian; VAROQUAUX, Gaël; GRAMFORT, Alexandre; MICHEL, Vincent; THIRION, Bertrand; GRISEL, Olivier; BLONDEL, Mathieu; PRETTENHOFER, Peter; WEISS, Ron; DUBOURG, Vincent; VANDERPLAS, Jake; PASSOS, Alexandre; COURNAPEAU, David; BRUCHER, Matthieu; PERROT, Matthieu; DUCHESNAY, Édouard. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*. 2011, roč. 12, č. 85, s. 2825–2830. Dostupné také z: http://jmlr.org/papers/v12/pedregosa11a.html.

14. *Welcome to Python.org* [online] [cit. 2023-04-08]. Dostupné z: https://www.python.org/.

15. *Project Jupyter | Home* [online] [cit. 2023-04-08]. Dostupné z: https://jupyter.org/.

16. *Scrapy | A Fast and Powerful Scraping and Web Crawling Framework* [online] [cit. 2023-04-08]. Dostupné z: https://scrapy.org/.

17. *WikiCFP : Call For Papers of Conferences, Workshops and Journals* [online] [cit. 2023-04-08]. Dostupné z: http://www.wikicfp.com/cfp/.

18. *Chrome DevTools* [online] [cit. 2023-04-19]. Dostupné z: https://developer.chrome.com/docs/devtools/.

19. *Beautiful Soup Documentation — Beautiful Soup 4.4.0 documentation* [online] [cit. 2023-04-09]. Dostupné z: https://beautiful-soup-4.readthedocs.io/en/latest/.

20. CHEN, D.Y. *Pandas for Everyone: Python Data Analysis*. Pearson Education, 2017. Addison-Wesley Data & Analytics Series. ISBN 9780134547053. Dostupné také z: https://books.google.cz/books?id=7zhDDwAAQBAJ.

21. *Polars* [online] [cit. 2023-04-09]. Dostupné z: https://www.pola.rs/.

22. *Matplotlib — Visualization with Python* [online] [cit. 2023-04-09]. Dostupné z: https://matplotlib.org/.

23. *Seaborn: statistical data visualization — seaborn 0.12.2 documentation* [online] [cit. 2023-04-09]. Dostupné z: https://seaborn.pydata.org/.

24. *Plotly Python Graphing Library* [online] [cit. 2023-04-19]. Dostupné z: https://plotly.com/python/.

25. LOPER, Edward; BIRD, Steven. *NLTK: The Natural Language Toolkit*. 2002. Dostupné z arXiv: cs/0205028 [cs.CL].

26. ŘEHŮŘEK, Radim; SOJKA, Petr. Software Framework for Topic Modelling with Large Corpora. In: *Proceedings of LREC 2010 workshop New Challenges for NLP Frameworks* [paměťový nosič]. Valletta, Malta: University of Malta, 2010, s. 46–50. ISBN 2-9517408-6-7. Dostupné také z: http://www.fi.muni.cz/usr/sojka/presentations/lrec2010-poster-rehurek-sojka.pdf.

27. ŘEHŮŘEK, Radim. *Semantic-based plagiarism detection*. Brno, 2008. Dostupné také z: https://is.muni.cz/th/qbatd/.

28. ABADI, Martín. TensorFlow: Learning Functions at Scale. In: *Proceedings of the 21st ACM SIGPLAN International Conference on Functional Programming*. Nara, Japan: Association for Computing Machinery, 2016, s. 1. ICFP 2016. ISBN 9781450342193. Dostupné z DOI: 10.1145/2951913.2976746.

29. *Keras: Deep Learning for humans* [online] [cit. 2023-04-11]. Dostupné z: https://keras.io/.

30. *PyTorch* [online] [cit. 2023-04-11]. Dostupné z: https://pytorch.org/.

31. *Hugging Face – The AI community building the future.* [Online] [cit. 2023-04-12]. Dostupné z: https://huggingface.co/.

32. WOLF, Thomas; DEBUT, Lysandre; SANH, Victor; CHAUMOND, Julien; DELANGUE, Clement; MOI, Anthony; CISTAC, Pierric; RAULT, Tim; LOUF, Remi; FUNTOWICZ, Morgan; DAVISON, Joe; SHLEIFER, Sam; PLATEN, Patrick von; MA, Clara; JERNITE, Yacine; PLU, Julien; XU, Canwen; LE SCAO, Teven; GUGGER, Sylvain; DRAME, Mariama; LHOEST, Quentin; RUSH, Alexander. Transformers: State-of-the-Art Natural Language Processing. In: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*. Online: Association for Computational Linguistics, 2020-10, s. 38–45. Dostupné z DOI: 10.18653/v1/2020.emnlp-demos.6.

33. *Streamlit • The fastest way to build and share data apps* [online] [cit. 2023-04-12]. Dostupné z: https://streamlit.io/.

34. *MongoDB: The Developer Data Platform | MongoDB* [online] [cit. 2023-04-12]. Dostupné z: https://www.mongodb.com/.

35. KUC, R.; ROGOZINSKI, M. *Elasticsearch Server*. Packt Publishing, 2013. Community experience distilled. ISBN 9781849518444. Dostupné také z: https://books.google.cz/books?id=PEFK3MuwBsIC.

36. *Open Distro for Elasticsearch* [online] [cit. 2023-04-12]. Dostupné z: https://opendistro.github.io/for-elasticsearch/.

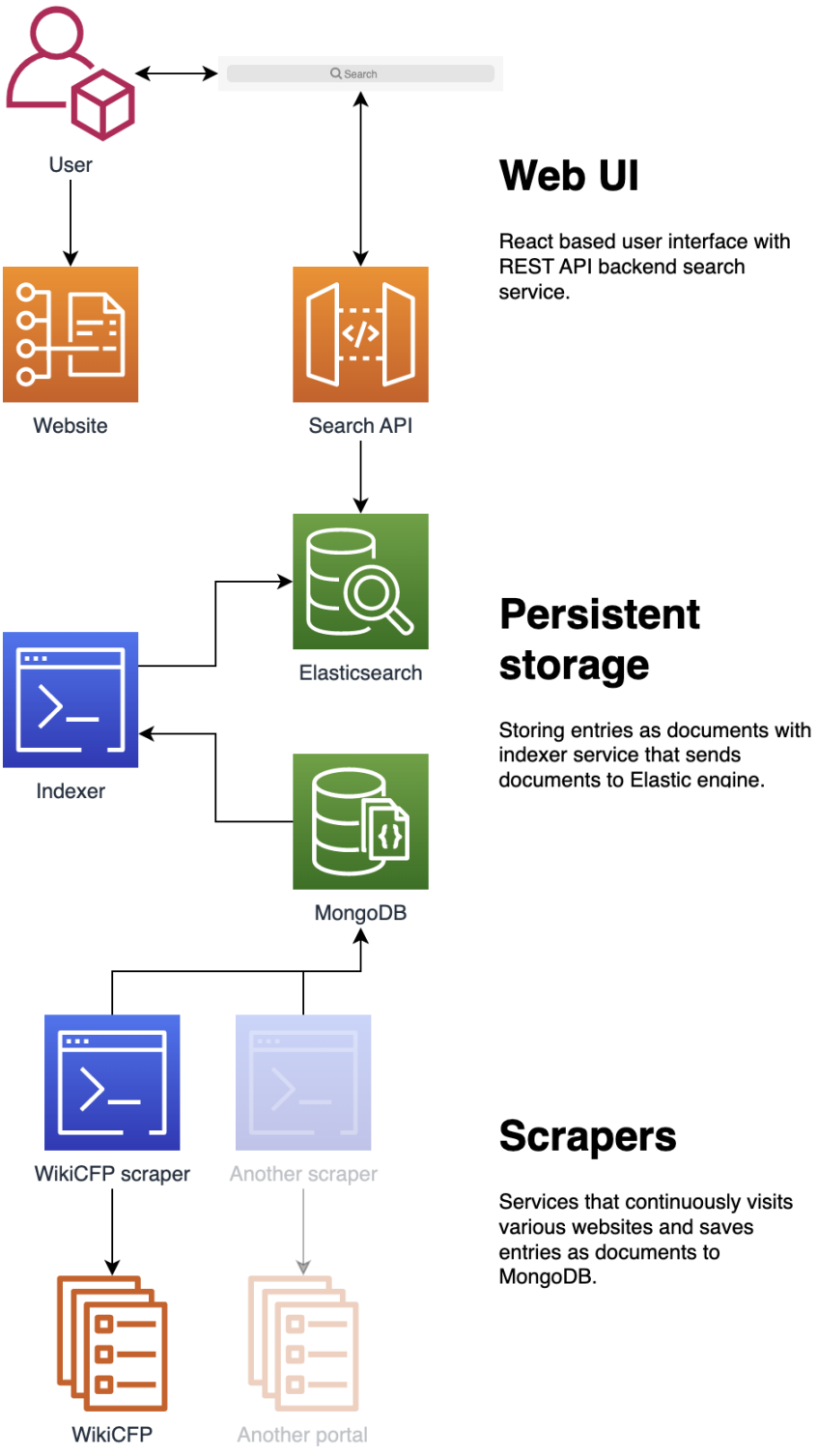
37. *K-NN - Open Distro Documentation* [online] [cit. 2023-04-12]. Dostupné z: https://opendistro.github.io/for-elasticsearch-docs/docs/knn/.

38. *Understanding searches better than ever before* [online] [cit. 2023-04-13]. Dostupné z: https://blog.google/products/search/search-language-understanding-bert/.

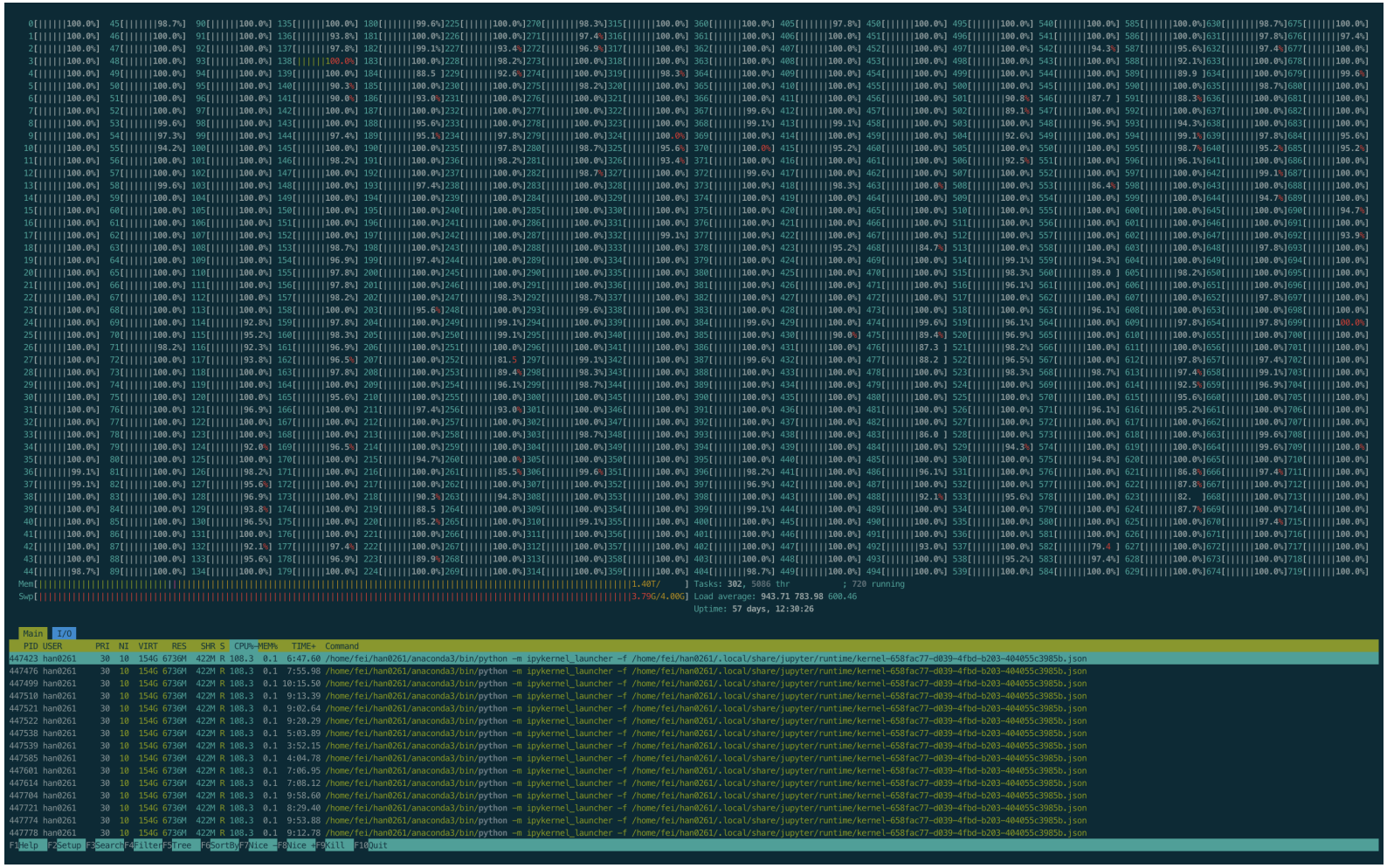
39. *Vyhledávání pomocí významových vektorů - Blog Seznam.cz* [online] [cit. 2023-04-13]. Dostupné z: https://blog.seznam.cz/2021/02/vyhledavani-pomoci-vyznamovych-vektoru/.

40. *Train and run machine learning models faster | Cloud TPU* [online] [cit. 2023-04-13]. Dostupné z: https://cloud.google.com/tpu.

41. *HPE Superdome Flex Servers* [online] [cit. 2023-04-14]. Dostupné z: `https://www.hpe.com/us/en/servers/superdome.html`.




Obrázek 2: Architektura aplikace




Obrázek 3: Ukázka obrazovky nástroje htop na serveru argexpr3 v průběhu XGBoost experimentů