VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV INFORMAČNÍCH SYSTÉMŮ
DEPARTMENT OF INFORMATION SYSTEMS

METODY KLASIFIKACE TEXTU V KONTEXTU WEBOVÝCH STRÁNEK

TEXT CLASSIFICATION METHODS IN THE CONTEXT OF WEB PAGES

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — **Bc. PATRIK TRSTENSKÝ**
AUTHOR

VEDOUCÍ PRÁCE — **doc. Ing. RADEK BURGET, Ph.D.**
SUPERVISOR

BRNO 2023

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

Zadání diplomové práce

147036

Ústav:	Ústav informačních systémů (UIFS)
Student:	**Trstenský Patrik, Bc.**
Program:	Informační technologie a umělá inteligence
Specializace:	Informační systémy a databáze
Název:	**Metody klasifikace textu v kontextu webových stránek**
Kategorie:	Umělá inteligence
Akademický rok:	2022/23

Zadání:

1. Prostudujte existující knihovny a nástroje pro klasifikaci textu a rozpoznání pojmenovaných entit (named entity recognition - NER) a způsob jejich trénování pro konkrétní cílovou doménu.
2. Seznamte se s možnostmi publikace strukturovaných dat v rámci webových stránek se zaměřením na technologie JSON-LD a RDFa. Prostudujte existující RDF znalostní báze.
3. Po konzultaci s vedoucím navrhněte sadu nástrojů pro tvorbu datové sady vhodné pro trénování klasifikátorů textu pomocí publikovaných strukturovaných dat nebo s využitím znalostních bází.
4. Implementujte navržené nástroje a vytvořte datové sady minimálně pro dvě cílové domény. Jejich volbu konzultujte s vedoucím. Proveďte trénování zvoleného klasifikátoru pomocí získaných dat.
5. Vyhodnoťte vlastnosti natrénovaného klasifikátoru.
6. Zhodnoťte dosažené výsledky.

Literatura:

- Xiang Cheng et al.: An End-to-End Solution for Named Entity Recognition in eCommerce Search, AAAI IAAI-2021 Highly Innovative Applications
  https://arxiv.org/abs/2012.07553
- Dále dle pokynů vedoucího.

Při obhajobě semestrální části projektu je požadováno:
Body 1 až 3.

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

Vedoucí práce:	**Burget Radek, doc. Ing., Ph.D.**
Vedoucí ústavu:	Kolář Dušan, doc. Dr. Ing.
Datum zadání:	1.11.2022
Termín pro odevzdání:	17.5.2023
Datum schválení:	26.10.2022

Abstrakt

Tato práce se zabývá problematikou klasifikací textu v kontextu webových stránek. Zkoumá dostupné metody klasifikace a jejich přesnost nad čistým textem z webové stránky. Zabývá se sestrojením datasetu pro trénování těchto metod pro konkrétní doménu. Data pro vytvoření datasetu získáváme z veřejně dostupných stránek, které využívají RDF dokumentů zadefinovaných v HTML kódu. Závěr práce sestává z vytvoření dvou datasetů pro dvě různé domény, dále z využití těchto datasetů na trénování modelů a následného testování jejich přesnosti.

Abstract

This work deals with the issue of text classification in the context of websites. It examines available classification methods and their accuracy over web page plain text. It deals with constructing a dataset for training these methods for a specific domain. We obtain data for creating the dataset from publicly available websites that utilize RDF documents defined in HTML code. The conclusion of the work consists of the creation of two datasets for two different domains. Furthermore, the use of these datasets for training models and testing of their accuracy.

Klíčová slova

klasifikace textu,rozpoznávání pojmenovaných entit, dolování informací z webu, Python, Sémantický web, JSON-LD, RDF, DBPedia

Keywords

text clasification, named entity recognition, web mining, Semantic web, Python, JSON-LD, RDF, DBPedia

Citace

TRSTENSKÝ, Patrik. *Metody klasifikace textu v kontextu webových stránek*. Brno, 2023. Diplomová práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce doc. Ing. Radek Burget, Ph.D.

Metody klasifikace textu v kontextu webových stránek

Prohlášení

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

.....
Patrik Trstenský
16. května 2023

Poděkování

Touto cestou bych chtěla velmi poděkovat vedoucímu mé diplomové práce, panu docentovi Ing. Radokvi Burgetovi, Ph. D, za vstřícnost, podporu, věcné připomínky a velmi cenné rady při konzultacích.

Obsah

Seznam obrázků

Kapitola 1

Úvod

V dnešní době se nám naskytují různé možnosti dolování informací z webu. Zpočátku se využívaly různé parsery, které nebyly až tak efektivní a generické pro různé druhy webových stránek. Postupně se tak začaly využívat různé metody strojového učení ke zvýšení efektivity získávání dat. Většina metod se však zaměřuje na kód stránky, případně vizuální vlastnosti, ze kterých získává data.

Pro výše uvedené důvody se tedy chceme v naší práci zaměřit na jiný přístup získávání informací z webů. Namísto získávání znalostí z kódu nebo vzhledu stránky chceme získávat informace pouze na základě čistého textu webové stránky. Momentálně existují metody, jak získávat informace ze souvislého textu. V takovém textu máme dostatečný kontext pro detekci informací. Avšak text z webové stánky nám takový kontext neposkytuje. Často obsahuje osamocená slova bez okolního kontextu, jejich detekce je tudíž náročnější.

V naší práci se proto zaměříme na ověření, zda je vůbec možné rozpoznat výskyty pojmenovaných entit jako název produktu, jména osob apod., a to pouze na základě kontextu, který nám poskytne text z webové stránky. Další hlavní otázkou, na kterou se budeme snažit najít odpověď, je s jakou přesností jsme schopni detekovat tyto entity.

Prvním krokem k tomu, abychom ověřili stanovené cíle, bude zkoumat aktuální nástroje, které nabízejí metodu rozpoznávání pojmenovaných entit z textu. Na něj bude navazovat vytvoření datasetů pro zvolenou klasifikační metodu. Proto se budeme věnovat způsobu, odkud a jak je možné získat dostatečné množství dat. Konkrétně se zaměříme na sémantický web, který obsahuje obrovské množství veřejně dostupných znalostních bází a také RDF dokumentů definovaných přímo v HTML kódu. Ten bude popisovat způsob získávání dat a vytváření datasetů, který si popíšeme v kapitole návrhu a implementace. Na závěr budeme trénovat naše modely pomocí vytvořených datasetů. Provedeme tedy experimenty a budeme porovnávat přesnosti jednotlivých metod. Poté vyvodíme závěr a dosavadní výsledky zhodnotíme.

Kapitola 2

Zpracování přirozeného jazyka v textu

V rámci této kapitoly si shrneme, co znamená *zpracování přirozeného jazyka v textu*, proč se touto problematikou zabýváme, dále si ukážeme najčastější metody pro zpracování přirozeného jazyka v textu a také představíme již existující nástroje, které se používají.

2.1 Zpracování přirozeného jazyka v textu obecně

Nejprve se seznámíme s pojmem NLP neboli **N**ature **L**anguage **P**rocessing a poté si více rozvineme jeho specifikaci na text. NLP je podoblast lingvinistiky, informatiky a umělé inteligence, jež se snaží dosáhnout toho, aby počítače rozuměly výrokům a slovům napsaným v lidském jazyce. NLP tedy zahrnuje vývoj algoritmů a modelů, které dokáží analyzovat, pochopit a generovat lidský jazyk. Taktéž může být použit k vytváření systémů založených na jazyku jako jsou chatboti, rozpoznávání řeči, překladače atd.

NLP má opravdu široký záběr, ale v naší práci se omezíme pouze na zpracování textů, ze kterého se budeme snažit dolovat důležité informace. Představíme si některé hlavní úkoly, jimiž se NLP zabývá pro získávání informací z textů:

1. **Jazykové modelování:** snažíme se předpokládat další slovo na základě předchozí sekvence slov, užitečné při překládání textů nebo vygenerování jeho shrnutí.

2. **Rozpoznávání pojmenovaných entit:** identifikace a značkování pojmenovaných entit v textu, jako například osoba, organizace, místo. Této metodě se budeme věnovat více v kapitole 2.3.

3. **Klasifikace textu:** jedná se o přiřazení textu do jedné nebo více kategorií na základě jejího obsahu.

4. **Značkování slovních druhů:** značkování slov ke spravným slovním druhům (např. pods. jméno, přid. jméno, sloveso atd.

5. **Sumarizace textů:** extrakce souhrnu z textu, například počet výskytů daného slova v textu.

2.2 Metody klasifikace textu

V této kapitole popíšeme základní metody, které se používají pro klasifikaci textu. Těmto metodám se věnujeme, abychom uměli lépe pochopit fungování zvolených nástrojů, jenž jsou popsány v kapitole 2.3. Předzpracování dat zmíníme jen okrajově, jelikož jsou detailněji popsány až v části implementace.

2.2.1 Naivní Bayesův klasifikátor

Bayesův klasifikátor je statistický klasifikátor, který například určuje, zda konkrétní dokument patří do dané třídy. Je to učení se s učitelem, tudíž na vstupu očekáváme už přiřazená data, podle kterých budeme klasifikátor učit. Naivním jej označujeme z důvodu předpokladu nezávislosti mezi jednotlivými atributy.

Máme nezařazený dokument, D jenž se skládá z příznakového vektoru $D = \langle d_1, d_2, .., d_n \rangle$ a trénovací data, M která se také skládají z příznakových vektorů. Dále máme množinu tříd $C = \{c_1, c_2, .., c_m\}$, do kterých chceme klasifikovat dokument. Naším úkolem bude vypočítat pravděpodobnost, $P(C/D)$ tedy, že nezařazený dokument patří do dané třídy. Obráceně pak máme *posteriorní* pravděpodobnost $P(D/C)$, která tvrdí, že dostaneme nezařazený dokument ve třídě c. Poslední důležitá pravděpodobnost *priorní* je zapsána jako $P(C)$, která vyjadřuje pravdepodobnost dané třídy. Nyní si představíme Bayesův teorém, podle kterého budeme pracovat.

Máme A a B jakožto dva náhodné jevy a jejich pravděpodobnosti jsou $P(A)$ a $P(B)$, a pokud $P(B) > 0$, potom platí:

$$P(A/B) = \frac{P(B/A)P(A)}{P(B)} \tag{2.1}$$

Doplníme proměnné podle našeho zadaného příkladu a dostaneme:

$$P(C/D) = \frac{P(D/C)P(C)}{P(D)} \tag{2.2}$$

Samozřejmě se snažíme dokument přiřadit do třídy, do níž spadne s největší pravděpodobností:

$$c_i = \mathbf{arg\,max}\, P(c/D) \tag{2.3}$$

Dále nám již zůstává dopočítat zbývající pravděpodobnosti. Jelikož $P(D)$ je konstantní pro všechny třídy, můžeme si ji dovolit zanedbat. Pravděpodobnost $P(c_i)$ můžeme získat z trénovacích dat:

$$P(c_i) = \frac{|M_{c_i}|}{|M|} \tag{2.4}$$

Nakonec pro získání pravděpodobnosti $P(D/c_i)$ použijeme náš naivní předpoklad nezávislosti atributu a dostaneme:

$$P(D/c_i) = \prod_{j=1}^{n} P(d_j/c_i) \tag{2.5}$$

Takovým způsobem dokážeme klasifikovat nový dokument do dané třídy na základě podmíněné pravděpodobnosti. V této kapitole jsem čerpal z těchto zdrojů: [4] [23]

2.2.2 Metoda podpůrných vektorů (SVM)

Metoda podpůrných vektorů, anglicky **S**upport **V**ector **M**achines, je další klasifikační metodou založenou na učení s učitelem. Základní myšlenkou je nalézt nejlepší hranici nebo nadrovinu, která odděluje různe třídy textu v trénovacích datech. Hranici se snažíme zvolit tak, abychom maximalně oddělili různé třídy a zároveň maximalizovali rozpětí, což je vzdálenost mezi hranicí a nejbližšími datovými body z každé třídy, viz obrázek 2.1. Když jsme úspěšně našli tuto hranici, nový text klasifikujeme tak, že uvidíme, do které strany hranice padne. Informace pro tuto kapitolu byly převzaty z [19] [11]

Rozeberme si tuto metodu detailněji a podívejme se, jak zpracovává lineárně separovatelný problém a lineárně neseparovatelný problém.

Lineárně separovatelný problém

Tento problém je v rámci řešení nejjednodušší, jelikož dokážeme skupinu dat rozdělit lineárně. Mějme zadanou množinu trénovacích dat jako dvojici (x_i, y_i), kde x_i představuje vektor reprezentující jeden bod a y_i třídu, do které patří. Budeme uvažovat pouze o dvou třídách, čili $y_i \in \{-1, 1\}$. Jako první si uvedeme reprezentaci nadroviny:

$$w \cdot x - b = 0 \tag{2.6}$$

Kde w je váhový vektor v normálové formě, x je vstupní vektor a b je konstanta posunu. Rozpětí pak dokážeme vypočítat jako $\frac{2}{||w||}$, tuto vzdálenost chceme maximalizovat, což můžeme provést do následujícího optimalizačního problému:

$$min\frac{1}{2}||w||^2 \text{ vzhledem k } y_i(w \cdot x_i - b) \geq 1, \forall i \tag{2.7}$$

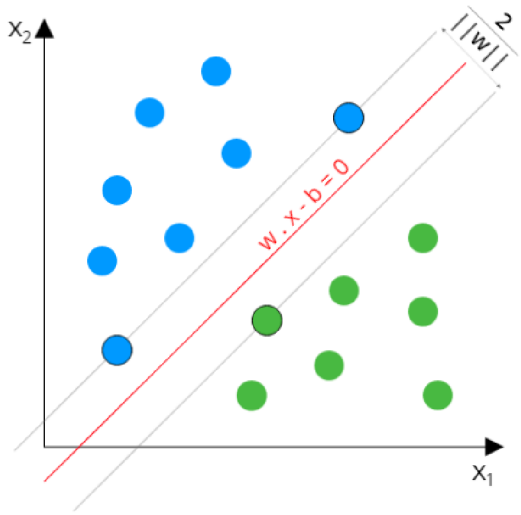
Jelikož se jedná o hledání vyhovujících parametrů pro w a b, musíme řešit kvadratický optimalizační problém, který lze zjednodušit za pomoci lagrangeové metody,

$$\sum_{i=1}^{n} a_i - \frac{1}{2}\sum_{i,j=1}^{n} a_i a_j y_i y_j (x_i \cdot x_j) \tag{2.8}$$

kde musí platit že:

$$0 \leq a_i, \forall_i \text{ a zároveň} \sum_{i=1}^{n} a_i y_i = 0 \tag{2.9}$$

Pomocí této úpravy se nám podařilo zbavit vstupního vektoru w a posunutí b a omezit se pouze na Lagrangeův koeficient, pro který platí podmínky (2.9).



Obrázek 2.1: Klasifikace Lineárně separovatelného problému

Lineární neseparovatelný problém

Lineární neseparovatelný problém je problém v případě, když neumíme nalézt žádnou hranici, níž bychom dokázali oddělit jednotlivá trénovací data. Pro tuto klasifikaci se používá modifikace předešlého algoritmu Soft margin. Tato modifikace spočívá v zadefinování chybové hranice ξ, pro kterou platí následující podmínky:

$$\begin{aligned} \xi &= 0 \text{ klasifikován správně} \\ 0 < \xi &< 1 \text{ klasifikován správně, ale nachází se uvnitř hranice} \\ \xi &\geq 1 \text{ klasifikován chybně} \end{aligned}$$

Dále definujeme konstantu C představující penalizaci. Pak bude minimalizační úloha vypadat následovně:

$$min\frac{1}{2}||w||^2 + C\,|\sum_{i=1}^{n}\xi_i\,|, \forall_i \tag{2.10}$$

Po použití lagrangianovy metody dostáváme totožnou duální formu (2.8) za podmínek:

$$0 \leq a_i \leq C, \forall_i \text{ a zároveň} \sum_{i=1}^{n} a_i y_i = 0 \tag{2.11}$$

kde samotný klasifikátor je vyjadřen:

$$f(x) = sgn(w \cdot x - b) \tag{2.12}$$

Funkce Signum vrací -1, pokud patří do první třídy a 1, pokud spadá do třídy druhé.

Nelineární problém

K řešení tohoto problému se používá mapování, kde se snažíme zadaný problém namapovat do prostoru s vyšší dimenzí. Ve vyšší dimenzi pak dokážeme řešit daný problém jako lineární. Mapovací funkci můžeme zapsat jako:

$$\Phi = R^d \rightarrow H \tag{2.13}$$

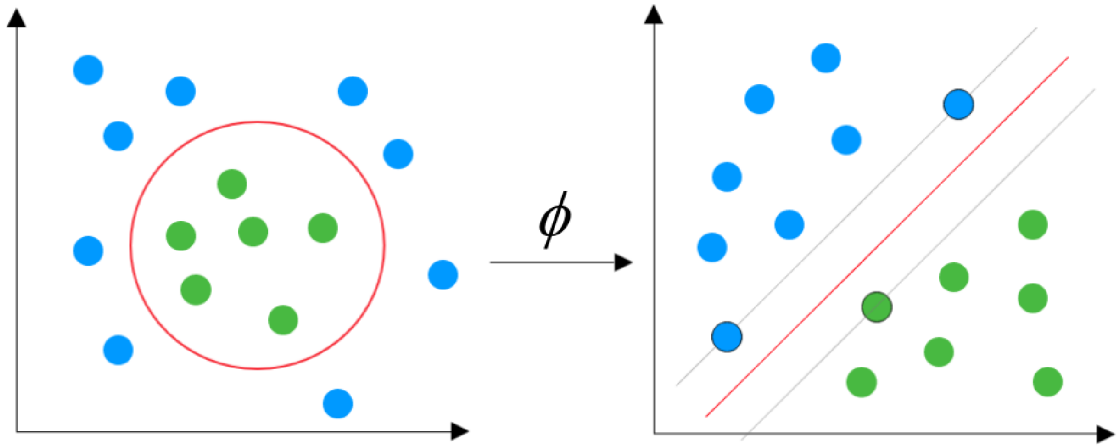
kde H je nový euklidovský prostor, jehož dimenze je větší než d. Potom vstupní trénovací vektory vypočítáme jako $\phi(x_i), \forall_i$. Duální problém bude pak vypadat následovně:

$$\sum_{i=1}^{n} a_i - \frac{1}{2} \sum_{i,j=1}^{n} a_i a_j y_i y_j (\phi(x_i) \cdot \phi(x_j)) \tag{2.14}$$

a rozhodovací funkci přeformulujeme na:

$$f_d(x) = sgn(w \cdot \phi(x) - b) \tag{2.15}$$

Pro lepší představivost je níže uveden ilustrační obrázek 2.2. Na psaní této podkapitoly jsem také čerpal z tohoto zdroje [15]



Obrázek 2.2: SVM mapovaný mezi prostory

2.2.3 Hloubkové učení

Modely hloubkového učení dosáhly kvalitních výsledků v mnoha oblastech, včetně širokého množství aplikace pro zpracování přirozeného jazyka. Hloubkové učení pro klasifikaci textu a dokumentů zahrnuje tři základní paralelní architektury hloubkového učení. Níže si podrobně popíšeme každý z těchto modelů. Informace na napsání této podkapitoly jsem čerpal z [9]

Hloubkové neuronové sítě (DNN)

Hloubkové neuronové sítě, anglicky také **D**eep **N**eural **N**etworks, jsou sítě navrženy tak, aby se učily vícenásobným propojením vrstev, kdy každá jedna vrstva pouze přijímá připojení z předchozí a zároveň poskytuje připojení pouze k další vrstvě ve skryté části. Obrázek 2.3

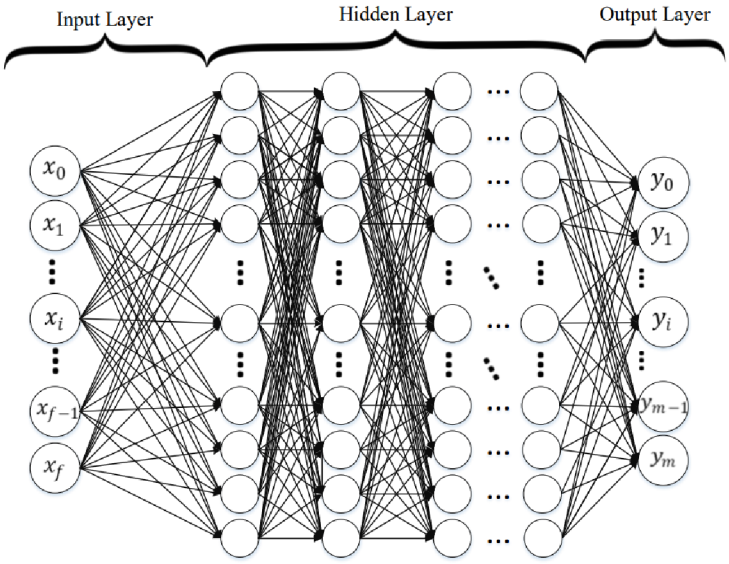
znázorňuje strukturu standartní DNN. Vstup se skládá ze spojení prostoru vstupních prvků s první skrytou vrstvou DNN. Vstupní vrstva může být vytvořena pomocí TF-IDF[1] vkládání slov nebo použití jiné extrakční metody. Výstupní vrstva je rovna počtu tříd pro vícetřídní klasifikaci, resp. pouze jedna pro binární klasifikaci. Implementace DNN je diskriminační model, který používá standartní algoritmus zpětného šíření *sigmoid* (2.16) nebo $ReLU$[2] (2.17) jako aktivační funkci. Výstupní vrstva pro vícetřídní klasifikace by měla být funkcí $Softmax$, jak je uvedeno v rovnici (2.18):

$$f(x) = \frac{1}{1 + e^{-x}} \in (0, 1) \tag{2.16}$$

$$f(x) = \max(0, x) \tag{2.17}$$

$$\sigma(z)_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_j}} \; \forall j \in \{1, ..., K\} \tag{2.18}$$

Máme-li soubor příkladů dvojice (x, y), kde $x \in X, y \in Y$, cílem je naučit se vztah mezi vstupním a cílovým prostorem za použití skryté vrstvy. V textové klasifikaci je vstupem často řetězec, který je generován vektorizací nezpracovaných textových dat.



Obrázek 2.3: Standartní, plně propojená Hloubková neuronová síť. [9]

Rekurentní neuronové sítě (RNN)

Rekurentní neuronové sítě, také **R**ecurrent **N**eural **N**etworks, jsou další architekturou neuronové sítě, která se také používá pro dolování a klasifikaci textu. RNN přiřadí předchozím datovým bodům více vah sekvence. Proto je tato technika výkonnou metodou pro textová, řetězcová a sekvenční data klasifikace. RNN zvažuje informace předchozích uzlů velmi sofistikovanou metodou, což umožňuje lepší sémantickou analýzu struktury datové sady. RNN

[1] https://jaketae.github.io/study/tf-idf/
[2] Rectified Linear Unit

většinou funguje pomocí LSTM nebo GRU pro klasifikaci textu, jak je znázorněno na obrázku 2.4, který obsahuje vstupní vrstvu (vkládání slov), skryté vrstvy a nakonec výstupní vrstvu. Tuto metodu lze formulovat takto:

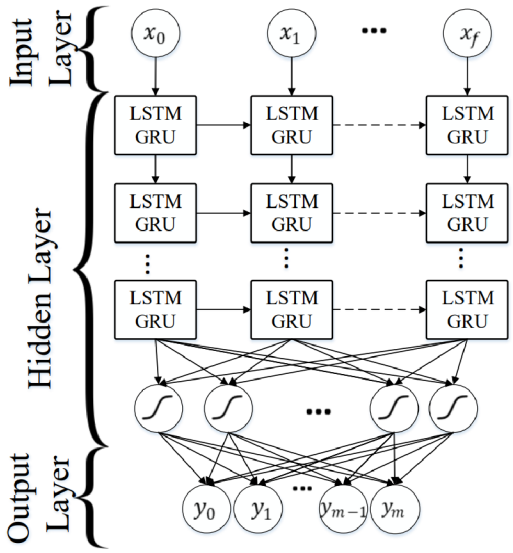
$$x_t = F(x_{t-1}, u_t, \theta) \tag{2.19}$$

kde x_t je stav v čase t a u_t odkazuje na vstup v kroku t. Přesněji řečeno můžeme použít váhy W jako parametry do rovnice (2.19):

$$x_t = W_{rec}\sigma(x_{t-1}) + W_{in}u_t + b \tag{2.20}$$

kde W_{rec} označuje opakující se váhu matice, W_{in} označuje vstupní váhy, b je *bias* a σ označuje elementární funkce.

Obrázek 2.4 znázorňuje rozšířenou architekturu RNN. Navzdory výše popsaným výhodám RNN je zranitelný vůči problémům *vanishing* a *exploding* gradientu. Když dojde k chybě, algoritmus sestupu gradientu je zpětně šířen sítí.



Obrázek 2.4: Standardní rekurentní neuronové sítě LSTM/GRU. [9]

Long short-term memory (LSTM)

LSTM zavedli S. Hochreiter a J. Schmidhuber a od té doby byl rozšířen mnoha způsoby.

LSTM je speciální typ RNN, který řeší problémy zachováním dlouhodobé závislosti, a to účinněji než základní RNN. LSTM je obzvlášť užitečný s ohledem na překonání problému *vanishing* gradient. Ačkoli LSTM má podobnou strukturu jako řetěz k RNN, LSTM používá několik bran k pečlivé regulaci množství informací, které jsou povoleny pro každý stav uzlu. Obrázek 2.5 ukazuje základní buňku modelu LSTM. Dále si vysvětlíme a popíšeme LSTM buňku:

$$i_t = \sigma(W_i[x_t, h_{t-1}] + b_i), \tag{2.21}$$
$$\tilde{C}_t = \tanh(W_c[x_t, h_{t-1}] + b_c), \tag{2.22}$$
$$f_t = \sigma(W_f[x_t, h_{t-1}] + b_f), \tag{2.23}$$
$$C_t = i_t \cdot \tilde{C}_t + f_t C_{t-1}, \tag{2.24}$$
$$o_t = z = \sigma(W_o[x_t, h_{t-1}] + b_o), \tag{2.25}$$
$$h_t = o_t \tanh(C_t) \tag{2.26}$$

kde rovnice (2.21) představuje vstup, rovnice (2.22) představuje objektivní hodnotu buňky, rovnice (2.23) definuje zapomínání buňky, rovnice (2.24) přepočítává novou hodnotu paměti, rovnice (2.25) a (2.26) definují konečnou hodnotu výstupu. Ve výše uvedeném popisu každé b představuje *bias* vektor, každé W představuje váhovou matici a x_t představuje vstup do paměťové buňky v době t. Dále i, c, f, o indexy odkazují na vstup, buněčnou paměť, zapomenutí a výstup. Obrázek 2.5 ukazuje grafické znázornění struktury.

Jestliže pozdější slova mají větší vliv než dřívější,výsledky RNN mohou být zkreslené. Konvoluční neuronové sítě na překonání tohoto zkreslení zavedly modely, které využívají *max-pooling* vrstvy pro určení diskriminačních frází v textových datech.

Gated Recurrent Unit (GRU)

GRU jsou zjednodušenou variantou architektury LSTM. GRU se však od LSTM liší, protože obsahuje dvě brány a GRU nemá vnitřní paměť. Vysvětlení GRU buňky:

$$z_t = \sigma_g(W_z x_t + U_z h_{t-1} + b_z) \tag{2.27}$$

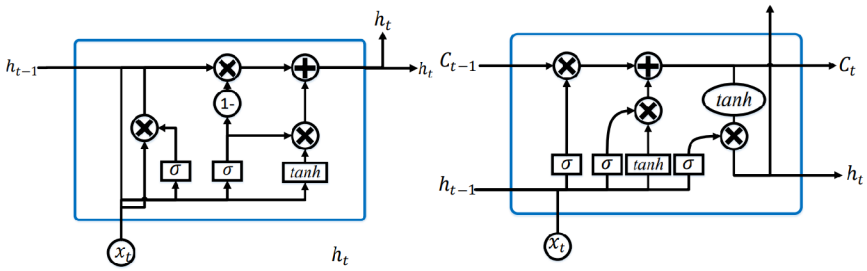
kde z_t odkazuje na aktualizační vektor, t, x_t znamená vstupní vektor, W, U a b představují parametr matice/vektory. Aktivační funkce (σ_g) je buď *sigmoid* nebo $ReLU$ a může být formulována následovně:

$$\tilde{r}_t = \sigma_g(W_r x_t + U_r h_{t-1} + b) \tag{2.28}$$

kde r_t představuje reset vektoru, t, z_t je aktualizace vektoru t.

$$h_t = z_t \circ h_{t-1} + (1 - z_t) \circ \sigma_h(W_h x_t + U_h(r_t \circ h_{t-1}) + b_h) \tag{2.29}$$

kde h_t je výstupní vektor, t a σ_h označuje funkci hyperbylického tangensu.

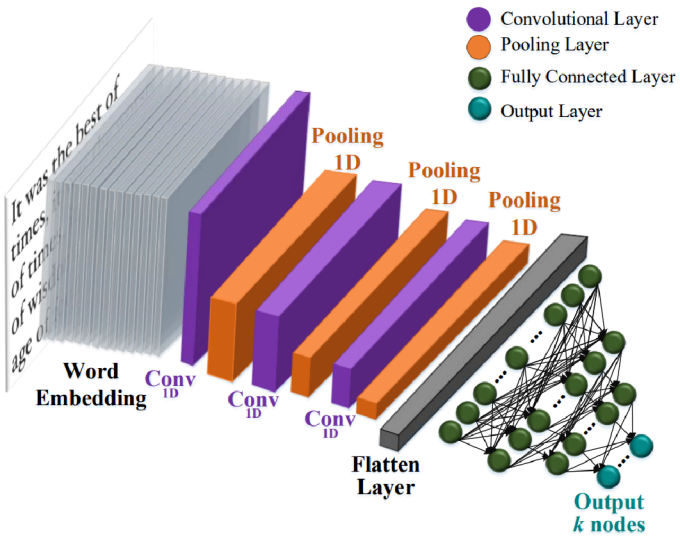


Obrázek 2.5: Nalevo je GRU buňka a napravo je LSTM buňka [9]

Konvoluční neuronové sítě (CNN)

Convolutional **N**eural **N**etwork je architektura hloubkového učení, která se běžně používá pro hierarchické třídění dokumentů. Ačkoli byl původně postaven pro zpracování obrazu, CNN byly také účinně použity pro klasifikaci textu. V základní CNN pro zpracování obrazu, obrazový tenzor je konvolvován se sadou *kernelů* o velikosti *dd*. Tyto konvoluční vrstvy se nazývají mapy prvků a lze je skládat tak, aby na vstupu poskytovaly více filtrů. Chcete-li snížit výpočetní složitost, CNN používají sdružování ke snížení velikosti výstupu z jedné vrstvy na další. Ke snížení výstupů při zachování důležitých vlastností se používají různé techniky sdružování.

Nejběžnější metodou sdružování je *max-pooling*, kde je vybrán maximální prvek z *poolingu*. Chcete-li přenést *pool* výstup z naskládaných map do další vrstvy, mapy jsou srovnány do jednoho sloupce. Finální vrstvy v CNN jsou obvykle plně propojeny. Obecně platí, že během kroku zpětného šíření konvoluční neuronové sítě jsou váhy a filtry detektoru upraveny. Potencionální problém, který vzniká při používání CNN pro klasifikace textu, je počet „kanálů“. Zatímco klasifikace obrázků aplikace má obecně málo kanálů (např. pouze 3 kanály RGB),pro aplikace klasifikace textu může být počet kanálů vicenásobně větší (např. 50 K), což má za následek velmi vysokou dimenzionalitu. Obrázek 2.6 ilustruje architekturu CNN pro klasifikaci textu, která obsahuje vkládání slov jako vstupní vrstvu 1D konvoluční vrstvy, 1D sdružovací vrstva, plně propojené vrstvy a nakonec výstupní vrstva.



Obrázek 2.6: Architektura konvoluční neuronové sítě (CNN) pro klasifikaci textu. [9]

2.2.4 Podmíněná náhodná pole (CRF)

Conditional **R**andom **F**ield je neorientovaný grafický model, který je znázorněný na obrázku 2.7. CRF je v podstatě kombinace výhod klasifikace a grafického modelování, které spojují schopnost kompaktního modelování vícerozměrných dat a schopnost využít prostor pro vysoce dimenzionální prvky predikce. Stav CRF je dán podmíněnou pravděpodobností sekvence Y za předpokladu, že nastala sekvence pozorování X, tj. $P(Y|X)$. CRF může začlenit složité rysy do sekvence pozorování, aniž by došlo k porušení nezávislosti předpo-

kladu modelováním podmíněné pravděpodobnosti sekvence než spojení pravděpodobnosti $P(X, Y)$. Klika (tj. plně propojený podgraf) jejíž potenciál se používá pro výpočet $P(X|Y)$. S ohledem na potenciální funkci pro každý klik v grafu pravděpodobnost a konfigurace proměnné odpovídá součinu řady nezáporných potenciálních funkcí. Hodnota vypočítaná každou potencionální funkcí je ekvivalentní pravděpodobností proměnných v odpovídající klice pro konkrétní konfiguraci, což je:

$$P(V) = \frac{1}{Z} \prod_{c \in klika(V)} \psi(c) \tag{2.30}$$

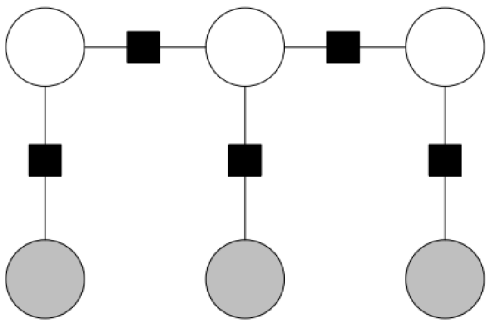
kde Z je normalizační term. Podmíněná pravděpodobnost $P(X|Y)$ může být formulována jako:

$$P(Y/X) = \frac{1}{Z} \prod_{t=1}^{T} \psi(t, y_{t-1}, y_t, X) \tag{2.31}$$

Vzhledem k potencionální funkci $\psi(t, y_{t-1}, y_t, X) = \exp(w \cdot f(t, y_{t-1}, y_t, X))$, podmínku pravděpodobnosti lze přepsat jako:

$$P(Y/X) = \prod_{t=1}^{T} \exp(w \cdot f(t, y_{t-1}, y_t, X)) \tag{2.32}$$

kde w je váhový vektor spojený s příznakovým vekterém vypočítaným pomocí f. Nejzřetelnější nevýhodou CRF je vysoká výpočetní náročnost trénovacího kroku, zejména pro soubory textových dat kvůli velkému počtu premenných. Navíc tento algoritmus nepracuje s neviditelnými slovy (tj. se slovy, která nebyla přítomna v trénovacích datech). Zdroje pro túto podkapitoli [9]



Obrázek 2.7: Podmíněné náhodné pole (CRF).Černé skříňky jsou přechodové kliky [9]

2.3 Nástroje pro klasifikaci textu

V této kapitole si představíme nástroje, které budeme používat na klasifikaci a získávání informací z textu. Detajlněji si povíme, co je to NER a jaké metody na něj níže uvedené nástroje využívaji. Bude to jeden z dúležitých způsobů, jimiž chceme extrahovat informace z webů.

Rozhodli jsme se zaměřit na již existující Python knihovny, které jsou volně dostupné. V jednotlivých podkapitolách si detailněji rozebereme, co daný nástroj nabízí a jaké metody

využívá na rozpoznávání pojmenovaných entit. Informace pro jednotlivé nástroje jsem čerpal z tohoto zdroje[21]

Rozpoznávání pojmenovaných entit (NER)

Rozpoznávání pojmenovaných entit anglicky **N**amed **E**ntity **R**ecognition je dílčím problémem získávání informací a zahrnuje zpracování strukturovaných i nestrukturovaných dokumentů a také identifikací výrazů, které se odkazují na lidi, místa organizace a společnosti. NER zahrnuje dva hlavní úkoly, první spočívá v identifikaci vlastních jmen v textu a druhý úkol spočíva v zařazení těchto jmen do předdefinovaných kategoriích zájmů jako jména osob, organizace (společnosti, vládní organizace, výbory atd.), místa (města, země, řeky atd.), výrazy pro datum a čas.

Pro lidi je NER intuitivně jednoduchý, protože mnoho pojmenovaných entit jsou vlastní jména a většina z nich mají počáteční velká píjsmena a lze je snadno rozpoznat, ale pro stroj je to náročné. Někdo by si mohl myslet, že pojmenované entity lze snadno klasifikovat pomocí slovníku, protože většina pojmenovaných entit je tvořena podstatnými jmény, což není správně. Jak čas plyne, vznikají nová vlastní podstatná jména. Proto je nemožné přidat všechna podstatná jména do slovníku.

Jestliže by se nám podařilo přidat také všechna podstatná jména do slovníku, je zde další zásadní problém, a to že není snadné rozhodnout jejich význam. Většina problémů v NER spočívá v tom, že tyto slova mají sémantickou dvojznačnost, která závisí od kontextu. Pro ilustraci, kdy je *The White House* organizace a kdy místo ? Kdy je *June* jméno osoby a kdy název měsíce. Samozřejmě, když známe kontext, tak se dokážeme rozhodnout snadno. Například u *He visited Bush at White House* víme, že se jedná o označení místa, zatímco zde *White House announced the list of ministry candidate* se jedná o organizaci.

Automatická extrakce vlastních jmen je užitečná pro řešení mnoha problémů, jako je strojový překlad, získávání informací, odpovídání na otázky a sumarizace. Klíčem ke zpracování otázky je identifikace dotazu (kdo, co, kdy, kde atd.). V některých případech stačí pouze odpověď ano či ne. Hlavním cílem je tedy rozpoznání pojmenovaných entit a extrakce do některých konkrétních kategorií z textu podle smyslu jména. [12]

2.3.1 NLTK

NLTK[3], přesněji **N**atural **L**anguage **T**oolkit, je knihovna pro symbolové a statistické zpracování přirozeného jazyka. Umožňuje provádět různé úlohy zpracování textu: segmentaci, tokenizaci, značkování slovních druhů atd. Obvykle se používá v procesu výuky studentů a provádění výzkumu. Modul NER využívá klasifikátor **maximální entropie** vyškolený na korpusu ACE (Automatic Content Extraction). Maximální entropii si popíšeme podrobněji a také si ukažeme, jak rozpoznat pojmenované entity pomocí tohoto nástroje.

Maximální entropie

Používá princip maximální entropie. Princip říká, že hledáme model, který bude splňovat všechna naše omezení a nebude přijímat žádné jiné předpoklady. Abychom mohli definovat omezení, musíme nejprve definovat vlastnost. Vlastnost je funkce v následujícím tvaru. Obvykle se používají binární, ale obecně je možné použít jakékoliv nezáporné funkce. Jako příklad si navrhneme následující funkci k vyjádření vztahu mezi třídou *NamedEntity* OSOBA a psaním velkých píjsmen slova x.

[3] https://www.nltk.org

$$f(x,y) = \begin{cases} 1 & \text{ak } y \text{ je OSOBA a } x \text{ začína s velkým píjsmenem} \\ 0 & \text{jinak} \end{cases} \tag{2.33}$$

Omezení je pak definováno jako rovnost středních hodnot pro daný prvek.

$$E_p(f_i(x,y)) = E_{\tilde{p}}(f_i(x,y)) \tag{2.34}$$

kde $E_{\tilde{p}}(f_i(x,y))$ je střední hodnota vlastnosti vypočítaná přes trénovací data a střední hodnota modelu je $E_p(f_i(x,y))$. Je zaručeno, že takový model existuje. Rozdělení *maximum-likelihood* je v následující formě:

$$p(y/x) = \frac{1}{Z(x)} \exp \sum_{i=1}^{n} \lambda_i f_i(x,y) \tag{2.35}$$

$$Z(x) = \exp \sum_{i=1}^{n} \lambda_i f_i(x,y) \tag{2.36}$$

$Z(x)$ je pouze normalizační faktor a zajišťuje, že $p(y/x)$ je rozdělení pravděpodobnosti. Parametry modelu jsou $\lambda_1 ... \lambda_n$. [8]

Pracování s knihovnou

Jelikož pracujeme s python knihovnami, instalace je jednoduchá. Framework jednoduše nainstalujeme pomocí příkazu:

- `pip install nltk`.

Dále si ukážeme, jak můžeme jednoduše identifikovat předdefinované pojmenované entity v textu, které má NLTK již dopředu definované.

```
import nltk
sentence = "At eight o'clock on Thursday morning Arthur didn't feel good."
tokens = nltk.word_tokenize(sentence)
tagged = nltk.pos_tag(tokens)
print(tagged)

OUTPUT:
[('At', 'IN'), ('eight', 'CD'), ('o', 'NNS'), ('‘', 'NNP'),('clock', 'NN'),
('on', 'IN'), ('Thursday', 'NNP'), ('morning', 'NN'), ('Arthur', 'NNP'),
('didn', 'NN'), ('‘', 'NNP'), ('t', 'NN'), ('feel', 'VB'),('good', 'JJ'),
('.', '.')]
```

Tabulka 2.1: Tabulka Zkratek

Značka	Význam
IN	předložka / podřadicí spojka
NNS	podstatné jméno množného čísla
NNP	vlastní podstatné jméno, jednotné číslo
NN	podstatné jméno, jednotné číslo
VB	sloveso, základní tvar
RB	příslovce
JJ	přídavné jméno

Tento příklad byl převzat z originální stránky knihovny, viz zdroj [13].

2.3.2 SpaCy

Další knihovnou je SpaCy[4], která také nabízí nástroje pro zpracování přirozeného jazyka. Mimo jiné také poskytuje čtyři dopředu natřenované modely pro detekci pojmenovaných entit.

- en_core_web_sm
- en_core_web_md
- en_core_web_lg
- xx_ent_wiki_sm

Vybereme si en_core_web_lg a popíšeme princip na kterém funguje. Tento model je trénovaný na *OntoNotes* což je velký korpus anotovaného textu. Model je trénován s přechodným přístupem podobným tomu, který nastínili Lample [10]. Nicméně, zatímco Lample navrhuje použití obousměrné Long, Short-Term Memory (LSTM) které jsme si již popisovali, SpaCy místo toho používá konvoluční vrstvy se zbytkovým spojením, normalizací vrstev a maxout-nelinearitu. Dalším důležitým aspektem modelu SpaCy je použití *bloom embeddings*. *Bloom embeddings* využívá hashovací trik ke kompresi velkého, ale řídkého vektoru do hustého vložení, které obsahuje stejná data, ač komprimovaná, více zde [17] . V případě SpaCy berou *Bloom embeddings* v úvahu vlastnosti sousedního textu, což dává každému kontextu slova jedinečné vložení. Zdroje pro tuto podkapitolu [2].

Příklad použití

Ješte před instalací se musíme ujistit, zda máme aktualizované všechny nástroje, které potřebujeme `pip`, `setuptools` a `wheel`. Případně je nainstalovat:

- `pip install -U pip setuptools wheel`
- `pip install -U spacy`

po úspěšném nainstalování SpaCy knihovny můžeme začít s jednoduchou ukázkou rozpoznávání pojmenovaných entit.

[4] https://spacy.io

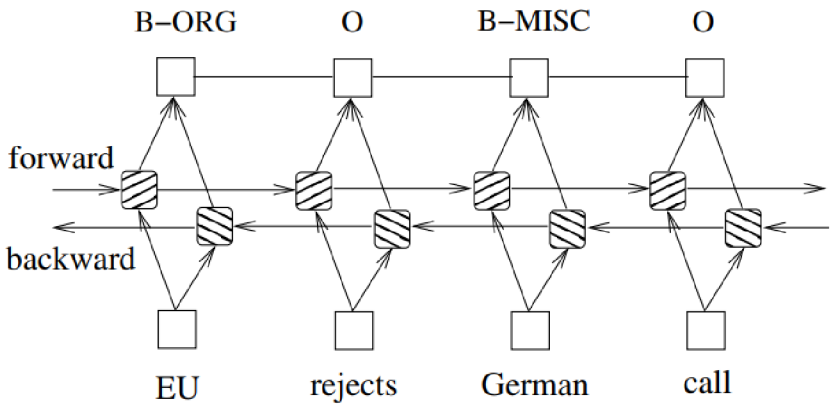
```
import spacy
from spacy import displacy

NER = spacy.load("en_core_web_lg")
sentence = "At eight o'clock on Thursday morning Arthur didn't feel good."
tagged = NER(sentence)

for word in tagged.ents:
    print(word.text,word.label_)

OUTPUT:
eight o'clock TIME
Thursday DATE
morning TIME
Arthur PERSON
```

2.3.3 Flair

Knihovna Flair[5] obsahující modely pro řešení úloh zpracování přirozeného jazyka, jako je rozpoznávání pojmenovaných entit, značkování slovních druhů a klasifikace. Vyvinutý týmem Zalando Research. Modul NER využívá kontextové vkládání získané pomocí obousměrného modelu znakového jazyka. Model s architekturou BI-LSTM-CRF se používá pro rozpoznávání pojmenovaných entit.

BI-LSTM-CRF

Jak vyplývá už ze samotného názvu, tento model kombinuje obousměrnou síť LSTM a síť CRF na vytvoření hybridního modelu BI-LSTM-CRF jehož architekturu můžeme vidět na obrázku 2.8. Kromě minulých vstupních hodnot a úrovně věty informace o tagu používané v modelu LSTM-CRF, model BILSTM-CRF může využívat budoucí vstupní hodnoty, které mohou zvýšit přesnost označování.



Obrázek 2.8: BI-LSTM-CRF model

[5] https://github.com/flairNLP/flair

Dále si popíšeme, jak funguje trénovací algoritmus pro model BI-LSTM-CRF které používá dopredný i zpětný tréningový postup. V každé epoše rozdělíme celá trénovací data do dávek a vždy zpracujeme jednu dávku. Každá dávka obsahuje seznam vět, která je určena parametrem velikosti dávky. Pro každou davku nejprve spustíme BI-LSTM-CRF dopredný přechod, který zahrnuje přechod oběma LSTM stavy (dopredny, zpětný). V důsledku toho získáváme výstupní skóre $f_\theta([x]_1^T)$ pro všechny značky na všech pozicích. Pak spustíme dopredný a zpětný přechod vrstvou CRF. Poté můžeme zpětne šířit chyby ze vstupu na vstup, což zahrnuje zpětný průchod pro zpětný a dopredný stav LSTM. Nakonec aktualizujeme síťové parametry, které zahrnují přechodovou matici $[A]_{i,j} \forall i, j$ a obousměrné LSTM parametry θ. [7]

Algorithm 1 obousměrný LSTM-CRF model trénování

```
for každou epochu do
    for každou dávku do
        1) BI-LSTM-CRF model dopredný přechod:
           - dopredný přechod pro dopredný stav LSTM
           - dopredný přechod pro zpětný stav LSTM
        2) dopředný a zpětný přechod CRF vrstvou
        3) BI-LSTM-CRF model zpětný přechod:
           - zpětný přechod pro dopredný stav LSTM
           - zpětný přechod pro zpětný stav LSTM
        4) aktualizace parametrů
    end for
end for
```

Příklad použití

Stejně jako v předchozích kapitolách, i zde si doinstalujeme knihovnu.

- `pip install flair`

pak jednoduše načítáme NER model a můžeme rozpoznávat entity.

```
from flair.data import Sentence
from flair.models import SequenceTagger

sentence = Sentence('I love Berlin .')
tagger = SequenceTagger.load('ner')
tagger.predict(sentence)
print(sentence)

OUTPUT:
Sentence: "I love Berlin ." -> ["Berlin"/LOC]
```

Kapitola 3

Strukturované informace na webu

V předchozí kapitole jsme se věnovali metodám klasifikace textu. Ukázali jsme si veřejně dostupné nástroje, které dokáží dolovat informace z textu. Všechny tyto nástroje, které jsme si ukázali, vznikaly podobným způsobem, tedy že měly již předem připravené datasety. Můžeme říci, že dataset je velmi důležitou součástí při vytváření modelů, zejména u těchto modelů založených na klasifikaci textu. Protože potřebujeme, aby nás model dobře klasifikoval, potřebujeme i dostatečně velký dataset. Samozřejmě tento dataset bude muset splňovat určite požadavky. Ale v této kapitole se budeme spíše soustředit na to, odkud tato již anotovaná data získat, abychom si je nemuseli anotovat a rozdělovat sami. V celé této kapitole si ukažeme možnosti, které dnešní technologie nabízejí. Zaměříme se zejména na sémantický web a ontologii.

3.1 Sémanticky web a Ontologie

Sémantický web je síť dat, která jsou spojena způsobem, že je mohou snadno zpracovat stroje místo lidí. Lze jej pojmout jako rozšířenou verzi stávající *World Wide Web* a představuje efektivní prostředek reprezentace dat ve formě globálně propojené databáze. Podporou začlenění sémantického obsahu do webových stránek se sémantický web zaměřuje na konverzi v současnosti dostupného webu nestrukturovaných dokumentů na web informací/dat.

Sémantický web řídí **W**orld **W**ide **W**eb **C**onsortium (W3C). Vychází z W3C **R**esource **D**escription **F**ramework (RDF) a je obvykle navržen se syntaxí, která k reprezentaci dat používá **U**niform **R**esource **I**dentifiers (URI). Tyto syntaxe jsou známé jako syntaxe RDF. Zahrnutí dat do souborů RDF umožňuje počítacožvým programům nebo webovým pavoukům vyhledávat, objevovat, shromažďovat, vyhodnocovat a zpracovávat data na webu.

Hlavním cílem sémantického webu je spustit vývoj stávajícího webu, aby uživatelé mohli vyhledávat, objevovat, sdílet a spojovat informace s menším úsilím. Lidé mohou používat web k provádění různých úkolů, jako je rezervace online vstupenek, vyhledávání různých informací, používání online slovníků atd. I tak nejsou stroje schopny provádět žádné z těchto úkolů bez lidského zásahu, protože webové stránky jsou vytvořeny tak, aby je mohli číst lidé, ne stroje. Sémantický web lze považovat za vizi budoucnosti, ve které by mohla být data rychle interpretována stroji, což by jim umožnilo provádět četné únavné úkoly související s objevováním, prolínáním a prováděním akcí na základě informací dostupných na webu.

Sémantický web je proces, který umožňuje strojům rychle porozumět komplikovaným lidským požadavkům a reagovat na ně s výhradou jejich významu. Tento druh porozumění

vyžaduje, aby příslušné informační zdroje byly sémanticky strukturovány, což je obtížný úkol.[20]

3.1.1 RDF

Jak jsme si již na úvodu této kapitoly načrtli, RDF je všeobecný framework na reptrezentaci vzájemně propojených dat na webu. RDF příkazy se používají k popisu a výměně metadat, což umožňuje standardizovanou výměnu dat na základě vztahů.

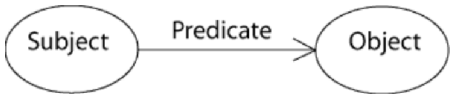
RDF se používá k integraci dat z více zdrojů. Příkladem tohoto přístupu je webová stránka, která zobrazuje online katalogy od výrobce a propojuje produkty s recenzemi na různých webech a s obchodníky prodávajícími produkty. Sémantický web je založen na použití rámce RDF k uspořádání informací na základě významů. Použitá literatura v kapitole [3] [16].

RDF trojice

Základní struktura jakéhokoli výrazu v RDF je sbírka trojic, z nichž se každá skládá zo subjektu, predikátu a objektu. Subjekt je vždy identifikovan jednoznačným identifikatorem URI. Subjekt je v podstatě zdroj, který je popisován. Predikát reprezentuje vlastnost, která popisuje daný subjekt a definuje vztah mezi subjektem a objektem. Objekt může byt literál, specifická datová hodnota jako řetězec, datum nebo číselná hodnota, nebo muže být definován jednoznačným identifikatorem URI. Jestliže je definován jako URI, může objekt vystupovat jako subjekt v další RDF trojici.

Tímto nám vznikne možnost, že se RDF trojice mohou zanorovat a tím definovat další vztahy. Tyto vztahy potom dokážeme reprezentovat v orientovaném grafu, který nazveme RDF graf. Potom jsou URI předměty značené jako elipsy, tudíž když máme subjekt nebo objekt definovaný URI. Je-li objekt definovaný jako literál, zaznačíme ho jako obdelník. Predikáty jsou orientované hrany, které spojují subjekty s objekty, jejichž směr je ze subjektu do objektu.

Na obrázku 3.1 vidíme jednoduchou ilustraci RDF trojice v RDF grafu, kde subjekt a objekt jsou definované URI.



Obrázek 3.1: RDF Trojice

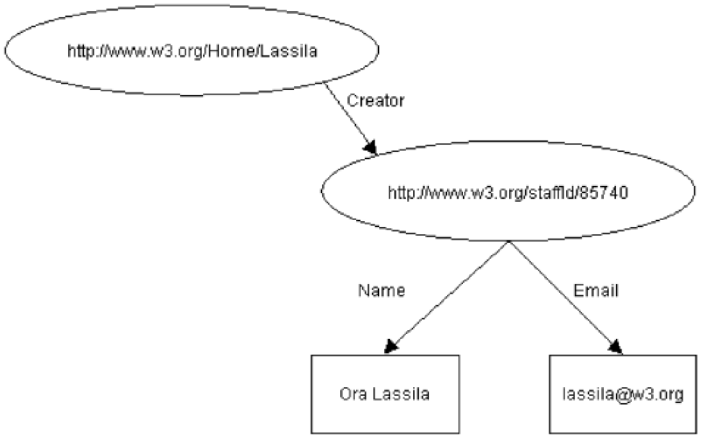
Príklad

Uvedeme si jednoduchý příklad, abychom si ukázali, jak je možné jednotlivé trojice skládat a zanorovat. Mějme následující věty:

"The individual referred to by employee id 85740 is named Ora Lassila and has the email address lassila@w3.org. The resource http://www.w3.org/Home/Lassila was created by this individual."

První věta nám říka, že pod ID zaměstnance 85740 je zamněstnanec jménem Ora Lassila s e-mailem lassila@w3.org. Druhá věta nám říka, že web http://www.w3.org/Home/Lassila byl vytvořený tímto zaměstnancem. Když si detailnější rozebereme první větu do RDF trojic,

tak zjistíme, že zamněstnanec bude <Subject>, kterému přiřazujeme dvě vlastnosti. E-mail a jméno jsou <Predikát> a nakonec "Ora Lassila"a "lassila@w3.org"budou definované jako <Objekt> v podobě literálu. U druhé věty to bude podobně, kde <Subjekt> je web s URI "http://www.w3.org/Home/Lassila", <Predikát> který nám určuje, kdo vytvořil daný <Objectt>, avšak zde bude určen ne jako literál, ale jako URI. Tím pádem dokážeme napojit dvě věty dohromady, čímž nám vznikne tento RDF model pro zadané věty:



Obrázek 3.2: RDF model pro zadané věty.

RDF Kontejner

Kontejner si můžeme představit jako seznam, kdy například potřebujeme jednomu projektu přiřadit více autorů. RDF definuje tři typy kontejnerů, které můžou představovat seznam zdrojů nebo literalů:

Bag

Jsou neuspořádané seznamy, například seznam tříd, kde není pořadí jmen studentů důležité. *Bags* nevynucují nastavenou sémantiku, takže hodnota se může v kontejneru objevit několikrát.

Sequence

jsou uspořádané seznamy. *Sequence* představují například dávku úloh odeslaných na zpracování, kde je důležité pořadí. Jako *Bags*, *sequence* umožňují duplicitní hodnoty.

Alternative

Nepovoluje duplicity. Můžeme použít tento kontejner na vyjádření alternativy, například zda se dopravíme do San Franciska pomocí letadla či auta. Aplikaci pak necháme rozhodnout, kterou alternativu zvolí.

RDF syntax

Příkazy RDF lze začlenit do webových stránek v jazyce HTML nebo uložit do samostatných souborů a propojit s daty ve webovém obsahu. Když bylo poprvé specifikováno RDF, příkazy RDF byly začleněny do dokumentů XML spojených s webovým obsahem.

Ačkoli prvním a jediným jazykem byl XML, postupem času se standardy pro kódování příkazů RDF rozšířily o následující tři syntaxe:

Turtle

Turtle[1] je nejoblíbenější textová syntaxe pro příkazy RDF. W3C jej popisuje jako „kompaktní a přirozenou textovou formu“, která obsahuje zkratky pro běžně používané vzory.

JSON-LD

JSON-LD[2] jako **J**avaScript **O**bject **N**otation for **L**inked **D**ata používá syntaxi JSON pro příkazy RDF.

N-Triples

N-Triples[3] je podmnožina syntaxe Turtle, navržená jako jednodušší textový formát pro příkazy RDF pro snadnější použití lidmi při psaní příkazů. Jednodušší formát také usnadňuje programům vytvářet a analyzovat RDF příkazy.

3.1.2 Ontologie

Ontologie je odvozena z filozofického oboru zvaného metafyzika, který se zabývá studií toho, co existuje. V informatice ontologie poskytuje rámec pro definování domény, která sestává ze souboru pojmů, charakteristik a vztahů.

Význam určitých informací je obecně vyjádřen na základě koncepčních informačních modelů, které se používají k modelování aplikací a strukturovaných údajů. Primární pojmy používané k vytváření takových modelů zahrnují entitu, činnost, prvek a účel. Konceptuální modely definují sémantické pojmy a mechanismy pro organizování informací vytvořením souboru předpokladů o skutečných aplikacích, které se mají modelovat. Například, pokud se předpokládá, že aplikace obsahuje vzájemně související entity, koncepční model definuje pojmy jako vlastnost a vztah.

Například Common Algebraic Specification Language je de facto standard v softwarové specifikaci, který je také považován za jazyk ontologie. Kóduje specifikace pro modularitu a strukturování softwaru s cílem zahrnout mnohé další existující specifikační jazyky.

OWL

Základní myšlenky sémantického webu a jazykové webové ontologie sahají přinejmenším do konce 50. let 20. století, kdy bylo představeno výpočetní zařízení **G**eneral **P**roblem **S**olver (GPS) jako jeden z prvních strojů, které dokázaly automaticky řešit dobře vytvořené logické vzorce. Principy umělé inteligence se pak postupně vyvíjely, aby po letech dále upevnily precedens pro aplikaci OWL přes web. **K**nowledge **R**epresentation (KR) se například snaží

[1] https://www.w3.org/TR/turtle
[2] https://json-ld.org
[3] https://www.w3.org/TR/n-triples

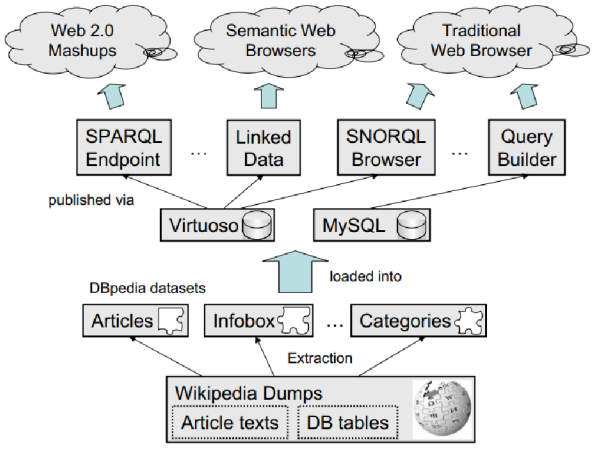
vytvořit systémy informací, které mohou stroje využívat způsobem, které jim umožní komplexní řešení problémů.

Jazyky webové ontologie jsou postaveny na standardu konsorcia World Wide Web nazvaném RDF. Rodina OWL se vyvinula tak, aby zahrnovala mnoho sytaxí a specifikací, a přitáhla velký zájem o obory, jako je medicína a akademická sféra.

3.1.3 DBpedia

Projekt DBpedia je veřejný projekt komunity lidí, kteří se snaží odvodit datový korpus z Wikipedie. Wikipedie je velmi rozsáhlou veřejně dostupnou doménou. Edice Wikipedie jsou dostupné ve více než 250 jazycích, a ten s anglickým jazykem představuje více než 1,95 milionu článků. Jako mnoho dalších webových aplikací, má Wikipedie problém v tom, že má omezené možnosti fulltextového vyhledávání, které umožňuje jen velmi omezený přístup k této cenné znalostní bázi. Jelikož Wikipedie dovoluje upravování dat: jsou rozporuplné, nekonzistentní,vyskytují se tam chyby a dokonce i spam. Projekt DBpedia se zaměřuje na úkol převádět obsah Wikipedie do strukturovaných znalostí, takže lze použít techniky sémantického webu, kladení sofistikovaných dotazů na Wikipedii a propojování s jinými datovými sadami, nebo vytváření nových aplikací.

Datové sady DBpedia lze importovat do aplikací třetích stran nebo k nim lze přistupovat online pomocí různých uživatelských rozhraní DBpedia. Obrázek 3.3 poskytuje přehled o procesu extrakce informací DBpedia a ukazuje, jak jsou extrahovaná data publikována na webu. Literatura pro celou podkapitolu [1]



Obrázek 3.3: Přehled komponentů DBPedie. [1]

Dataset

Dataset DBpedia v současné době poskytuje informace o více než 1,95 milionu „věcí“, včetně nejméně 80 000 osob, 70 000 míst, 35 000 albech, 12 000 filmů. Obsahuje 657 000 odkazů na obrázky, 1 600 000 odkazů na relevantní externí webové stránky, 180 000 externích odkazů na RDF databázi, 207 000 kategorií Wikipedie a 75 000 kategorií YAGO.

Dohromady se DBpedia skládá z přibližně 103 milionů RDF trojic. Dataset poskytuje ke stažení menší RDF soubory, které obsahují výše popsané informace.

Přístup

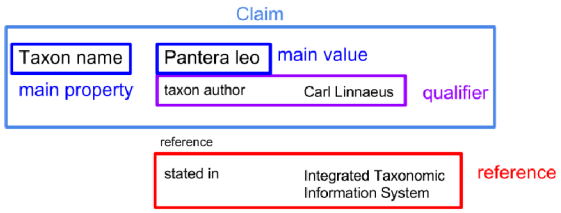
DBPedia poskytuje tři přístupové mechanismy k datasetům: Linked Data, SPARQL a soubory RDF ke stažení.

- *Linked Data* je metoda publikování dat RDF na webu, který se opírá o http:// URI identifikátor pro prostředek a protokol HTTP pro získaní zdrojů
- Posílání požadavku na API end-point SPARQL[4] .
- *N-Triple* serializace množin údajů jsou k dispozici ke stažení na webové stránce DBpedia a mohou být použity stránkami, které mají zájem o větší části souborových údajů.

3.1.4 WikiData

Wikidata je kolaborativní znalostní báze spuštěná v říjnu 2012 a hostitelem je Wikimedia Foundation. Wikidata jsou bezplatná a otevřená znalostní báze, kterou mohou číst a upravovat lidé i stroje. Wikidata fungují jako centrální úložiště pro strukturovaná data a jejich sesterských projektů Wikimedia včetně Wikipedie, Wikivoyage, Wikislovníku, Wikisource a dalších

Model Wikidata sa spolého na pojmy *item* a *statement*. *Item* představuje entitu, která ma identifikátor zvaný "qid"a může mít štítky nebo popisy ve více jazycích. *Statement* se skládá z nároku, může mít více nároků nebo také žádný. *Claim* se skládá z jedné hlavní struktury vlastnost-hodnota, což představuje nějaký fakt jako například „název taxonu je Pantera Leo“ a také může mít více volitelných *qualifier* jako „autorem taxonu je Carl Linné“. Na tomto obrázku si ukážeme příklad, který jsme si nyní popsali:



Obrázek 3.4: WikiData *Statement* ilustrovaný v příkladu [14]

Přístup

Přistupovat a dotazovat se nad Wikidata můžeme prostřednictvím klienta www.query.wikidata.org kde zadáváme dotaz v SPARQL jazyce nebo přes API call www.mediawiki.org/wiki/Wikibase/API. Použitá literatura [14].

[4] http://dbpedia.org/sparql

Kapitola 4

Návrh nástrojů

V této kapitole se budeme věnovat návrhu nástrojů, konkrétně způsobu získávání anotovaných dat z webů a jejich následnému anotování. Na závěr si porovnáme modely knihoven a vybereme tu, s níž budeme pracovat a ve které budeme vytvářet náš model.

4.1 Výběr zdroje dat

V předešlé kapitole jsme popisovali možnosti získávání anotovaných dat z webu. Na základě tohoto výzkumu jsme se rozhodli, že anotovaná data pro náš dataset budeme dolovat z webových stránek, jenž mají zadefinovaný RDF graf přímo ve zdrojích webové stránky. Nejprve jsme se snažili hledat webové stránky, které obsahují JSON-LD data definovaná v hlavičce nebo v těle dokumentu. Poté jsme chtěli sbírat data ze znalostních bází, avšak po delším zvažování jsme se rozhodli je nevyužít. Hlavním důvodem byl fakt, že k samotným anotovaným datům bylo zapotřebí kontextu, což nám tyto báze nedovolují. I když jsme zkoušeli selektovat názvy produktů s jejich popiskem, bohužel obsahoval jen stručné informace bez bližšího kontextu.

Po konzultaci s vedoucím práce jsme se dohodli, že budeme sbírat data pro oblast filmů a produktů.

4.2 Získání anotovaných dat

Prvním krokem tedy bude zjistit, zda daná webová stránka obsahuje strukturovaná data JSON-LD. V podstatě nám stačí stáhnout zdrojový kód stránky a najít element typu `<script>`, který bude definován pro JSON-LD data typ jako `{type="application/ld+json"}`. Tento způsob jsme si zvolili proto, jelikož v dnešní době spousta webů využívá možnosti jak zviditelnit svůj web a právě jednou z možností je definování JSON-LD nebo RDF dokumentů. Tyto dokumenty pak využívá Google algoritmus k zobrazování relevantních výsledků.

Protože JSON-LD může nabývat různých typů dat, budeme muset ověřit, zda jsou definována jako `Product`. K tomuto účelu slouží speciální atribut objektu `"@type"`. Dalším povinným atributem je `"name"`, který bude vždy obsahovat název produktu. Volitelné atributy, které nás budou zajímat pro náš dataset, jsou:

- `offers`: obvykle obsahuje informace o nabídce, ceně, dostupnosti, prodejci atd.
- `brand`: značka produktu

- `aggregateRating`: statistické informace o hodnocení produktu

Všechny atributy, které může produkt obsahovat, jsou definovány ve schématu[1]. Ukázka vytaženého JSON-LD objektu ze stránky www.alza.sk/EN:

```
{
    "@context": "http://schema.org",
    "@type": "Product",
    "name": "Samsung MS23F301TAS/EO",
    "description": "Microwave freestanding, volume 23l, number of power
    levels: 6, microwave heating performance 800W, plate size 28,8 cm,
    without grill,colour: stainless steel ",
    "image": "https://cdn.alza.sk/ImgW.ashx?fd=f5&cd=EAV2192u",
    "aggregateRating": {
        "@type": "AggregateRating",
        "ratingValue": "4.8",
        "ratingCount": "213"
    },
    "offers": {
        "@type": "Offer",
        "priceCurrency": "EUR",
        "price": "95.90",
        "url": "https://www.alza.sk/samsung-ms23f301tas-eo-d501637.htm",
        "itemCondition": "http://schema.org/NewCondition",
        "availability": "http://schema.org/InStock"
    },
    "brand": "Samsung",
    "sku": "EAV2192u",
    "mpn": "MS23F301TAS/EO",
    "gtin13": "EAV2192u"
}
```

Na základě analýzy atributů JSON-LD pro produkty jsme se rozhodli, že náš dataset bude obsahovat tyto typy anotací:

- **PRODUCT:** představuje název produktu
- **BRAND:** bude představovat značku daného produktu
- **PRICE:** reprezentace ceny produktu
- **RATING:** hodnocení produktu

Podobně jako u produktu budeme také pro filmovou doménu sbírat data z JSON-LD. Strukturální data filmů obsahují stejné povinné atributy jako u produktu objektu, kde je definován atribut `"@type"` , který v tomto případě nabývá hodnot `"Movie"` nebo `"TVSeries"`. Dalším společným povinným atributem je `"name"`, který v tomto případě definuje název filmu, seriálu a pod. Z volitelných atributů definovaných ve schématu[2] jsme vybrali tyto, které se vyskytují nejčastěji a zároveň je jich dostatečné množství pro přidání do našeho datasetu:

[1] https://schema.org/Product
[2] https://schema.org/Movie

- `actor`: seznam herců, kteří účinkovali ve filmu
- `genre`: seznam žánrů, do kterých je film přiřazen
- `director`: seznam režiserů, z jejichž dílny tento film pochází
- `aggregateRating`: statistické informace o hodnocení produktu

Z výše uvedených atributů budeme schopni vytvořit tato anotovaná data:

- **TITLE:** název filmu, seriálu apod.
- **GENRE:** žánr filmu, např. drama, komedie atd.
- **PERSON:** většinou se bude jednat o jména herců, režisérů apod.
- **RATING:** hodnocení filmu, seriálu apod.

Dalším krokem bude stáhnout stránky s produkty a vytáhnout z nich veškerý obsah jako čistý text. Následně budeme procházet tento text a všude, kde bude nalezena shoda, přiřadíme danou anotaci. To samé uděláme i pro vytváření datasetu pro filmy. Hlavním zdrojem byla dokumentace popisu JSON-LD objektu ze zdrojů [6] [5].

4.3 Výběr vhodné metody

V této podkapitole si porovnáme modely jednotlivých knihoven z kapitoly 2.3 a budeme se snažit vybrat ten s nejvyšší přesností. Samotné srovnání modelů je převzato z tohoto zdroje [22].

Budeme hodnotit výkon jednotlivých nástrojů, což znamená schopnost nástroje najít hranice pojmenované entity a správně určit její typ. Konkrétně budeme testovat dva typy hledání hranice, a to pro přesnou a částečnou shodu. Přesná shoda předpokládá přesnou shodu hranic předpovídaného a skutečného jména. Částečnou shodu předpovídaných a skutečných entit budeme hodnotit jako úspěšnou v takovém případě, pokud se typy shodují a hranice předpovídané entity jsou uvnitř hranic skutečné entity nebo naopak. Všechny modely byly trénovány na stejných korpusech, a to Kagge-2016[3] a CoNLL-2003[4]. Dále se budou porovnávat výsledky pro klasifikování tří typů (PER, ORG, LOC) a čtyř typů (PER, ORG, LOC, TIM), kde PER - název osoby, ORG - název organizace, LOC - místo a TIM indikátor času.

K vyhodnocení výkonnosti NER se používají následující metriky vypočítané s ohledem na typ i.

- **Precision:** podíl správně klasifikovaných entit ($TruePositive$) mezi všemi entitami přiřazenými klasifikátorům k typu i ($TruePositive + FalsePositive$)

$$P_i = \frac{TP_i}{TP_i + FP_i} \tag{4.1}$$

[3] https://www.kaggle.com/abhinavwalia95/entity -annotated-corpus
[4] https://www.clips.uantwerpen.be/conll2003/ner/

- **Recall:** podíl správně klasifikovaných entit ($TruePositive$) mezi všemi entitami patřícími do typu i ($TruePositive + FalseNegative$)

$$P_i = \frac{TP_i}{TP_i + FN_i} \tag{4.2}$$

- **F1-score:** harmonický průměr Precision a Recall

$$F1 = \frac{2 \cdot P_i \cdot R_i}{P_i + R_i} \tag{4.3}$$

Konečné hodnoty metrik se získají metodou makro-průměrování pro všechny typy podle vzorců:

$$P^M = \frac{\sum_{i=1}^{|C|} P_i}{|C|} \tag{4.4}$$

$$R^M = \frac{\sum_{i=1}^{|C|} R_i}{|C|} \tag{4.5}$$

$$F1^M = \frac{\sum_{i=1}^{|C|} F1_i}{|C|} \tag{4.6}$$

Tabulka 4.1: Výsledky experimentu NER pro přesnou shodu

Nástroje	**Model**	**F1-score**		
		Kaggle-2016		**CoNLL-2003**
		3 typy	**4 typy**	**3 typy**
NLTK	-	0.476	-	0.467
spaCy	en_core_web_sm	0.483	0.452	0.521
	en_core_web_md	0.503	0.468	0.570
	en_core_web_lg	0.496	**0.463**	0.597
	xx_ent_wiki_sm	0.501	-	0.597
Flair	-	**0.584**	-	**0.887**

Tabulka 4.2: Výsledky experimentu NER pro částečnou shodu

Nástroje	**Model**	**F1-score**		
		Kaggle-2016		**CoNLL-2003**
		3 typy	**4 typy**	**3 typy**
NLTK	-	0.616	-	0.555
spaCy	en_core_web_sm	0.649	0.619	0.608
	en_core_web_md	0.665	**0.631**	0.659
	en_core_web_lg	0.660	0.629	0.697
	xx_ent_wiki_sm	0.637	-	0.695
Flair	-	**0.691**	-	**0.904**

Jak můžeme vidět na výše uvedených tabulkách, knihovna Flair dosahovala nejvyšší přesnosti pro obě měření přesnosti. Samozřejmě, co se týče výsledků pro korpus, **CoNNL-2003** má Flair výrazně vyšší přesnost. Tento fakt je způsoben tím, že trénovací model byl

trénován na stejném korpusu, na kterém byl experiment proveden. Rozdíl mezi knihovnami a využívajícími architekturami jsme detailněji popisovali v kapitole 2.3. Na základě tohoto srovnání budeme pracovat s architekturou, kterou používá Flair, což je BI-LSTM-CRF.

Kapitola 5

Implementace

V této kapitole si popíšeme, jakým způsobem jsme získávali data a jak jsme řešili objevené komplikace. Popíšeme si vytvořený dataset a způsob anotování dat. Na závěr si po vytvoření datasetu implementujeme Model.

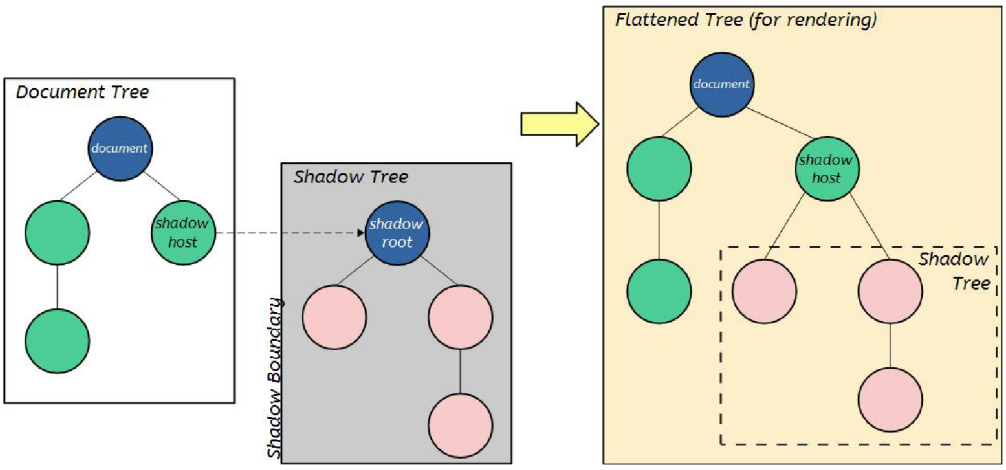
5.1 Sbírání dat

Na základě analýzy z předchozích kapitol budeme sbírat data z webových stránek, které obsahují JSON-LD objekty. Tyto objekty nám poskytují dostatečné množství informací k vytvoření anotovaných vět pro náš dataset.

Jako první jsme začali hledat webové stránky Produktů, kdy jsme se z počátku zaměřili na e-shopy s elektronikou. Hlavní důvod byl, že názvy s elektronikou jsou snadněji rozpoznatelné než produkty s méně výraznými názvy. Například *černý stůl*, kde nevíme, zda se jedná o produkt nebo o popis nábytku.

Pro získání obsahu stránky jsme nejprve používali python knihovnu `requests` kde jsme jednoduše použili metodu `requests.get(url,headers)`. Avšak po stažení produktů z prvního e-shopu jsme zjistili, že obsah stránky je poměrně chudší než originální stránky produktu. Proto jsme se rozhodli, že budeme sbírat obsah stránky pomocí knihovny `selenium` která se používá k testování webových stránek. Tato knihovna umožňuje přímé otevření webové stránky v prohlížeči, který je v automatizovaném módu. Díky této knihovně se nám podařilo získat dynamický text stránky a také odkliknout povolení soubory cookie, které v některých případech bylo nutné k dogenerování obsahu stránky. Také jsme si implementovali potřebné metody jako například `wait_for_element`, kde jsme čekali na načtení konkrétního elementu a pokud se do stanoveného limitu nenačetl, tak jsme pokračovali na další stránku. Důvod, proč jsme nevzali celou stránku, byl ten, že například některé produkty byly ve stavu nedostupné, nebo daný link byl již neplatný. Takové stránky pro nás neměly důležité informace, které by byly užitečné pro náš dataset. Otevírání každé stránky v automatizovaném okně bylo však výrazně pomalejší oproti prvnímu přístupu.

Dalším hlavním problémem, se kterým jsme se setkali, bylo získávání dat pro anotace typu RATING, ať už pro produkty nebo filmy. Důvodem tohoto problému bylo, že se html elementy nezkopírovaly po stažení obsahu stránky i poté, co jsme použili knihovnu `selenium`, což vyvstávalo z faktu, že některé stránky pro implementaci stejných částí stránky používaly takzvaný **Shadow DOM**. Shadow DOM umožňuje zapouzdření, tedy oddělení určité části elementů, značek a stylování od zbytku webové stránky. V podstatě to funguje tak, že se k prvku připojí skrytý oddělený DOM. Ilustrační obrázek níže.



Obrázek 5.1: Shadow DOM [18]

- **Shadow host:** běžný uzel DOM, ke kterému je připojen shadow DOM.
- **Shadow tree:** strom DOM uvnitř shadow DOM.
- **Shadow boundary:** místo, kde končí shadow DOM a začíná obyčejný DOM.
- **Shadow root:** kořenový uzel shadow stromu.

Na základě této skutečnosti jsme zjistili, že existuje způsob, kterým lze tento shadow DOM získat, a to spuštěním JavaScript kódu. Naštěstí otevíráme webové stránky v automatizovaném okně a knihovna `selenium` nabízí metodu, která spustí JavaScript kód nad danou stránkou. Potom jsme již jednoduše získali elementy ze shadow DOMu a vnořili je do elementu, který je měl obsahovat. Informace a definice byly převzaty ze zdroje [18]

Protože jsme chtěli dataset vytvořit pro anglický jazyk, většinu stránek jsme brali z domén `.uk`. Některé stránky poskytovaly i možnost definovat jazyk přímo v URL, např. `.cz/EN`, což bylo nejprve výhodou, ale po čase jsme zjistili, že některé stránky mají pouze částečný překlad. Toto zjištění vedlo ke skutečnosti, že jsme museli stažené stránky vyfiltrovat a odstranit.

Na konec jsme získali a uložili HTML kód stránek, který budeme v dalším kroku zpracovávat. Celkový seznam stránek a počet linků, ze kterých jsme stáhli obsah, je uveden v tabulkách 5.1 a 5.2.

5.2 Anotování dat a vytvoření datasetu

V předchozí kapitole jsme si ukazali, jak jsme stahovali celý obsah webových stránek. Dále si tedy popíšeme, jakým způsobem jsme je zpracovali do našich datasetů. Jako první jsme si načetli stažený soubor, který obsahoval HTML kód dané stránky. Po načtení obsahu souboru jsme použili knihovnu beautifullSoap, která dokáže načítat a zpracovat HTML kód do stromové struktury a následně s ní pracovat jako s objekty. Díky tomuto načtení jsme dokázali jednoduše najít všechna JSON-LD data pomocí příkazu `findAll('script', {'type': 'application/ld+json'})`, který nám vrátil seznam všech JSON-LD dát. Tento JSON-LD řetězec jsme převedli do JSON objektu, ve kterém jsme mohli jednoduše vyhledávat a najít potřebné atributy. Ukázka získaných dat z webové stránky po zpracování JSON-LD:

```
[('PRODUCT','Lenovo Legion M600s Qi Wireless Gaming Mouse'),
('BRAND','Lenovo'),
('PRICE','45.00'),
('RATING','4.4')]
```

Po získání anotovaných dat bylo nutné převést celý obsah stránky do čistého textu. Opět jsme si pomohli knihovnou `BeatufullSoap` která dokáže převést celý obsah stránky do textové podoby. Samozřejmě jsme samotný výstup funkce ještě upravovali. Prvním problémem bylo, že funkce každé ukončení elementu nahrazovala novým řádkem. Tímto způsobem vznikl text, který měl za sebou i deset znaků nového řádku a samotná slova byla od sebe vzdálena. V takovém textu jsme nebyli schopni vytvořit dostatečný kontext pro detekování pojmenovaných entit. Rozhodli jsme se, že celou stránku převedeme do textu a místo znaku nového řádku jsme použili mezeru. Velký počet mezer, které následovaly za sebou, jsme zredukovali na jednu. Tímto způsobem vznikl umělý kontext pro anotovaná data, který člověku nemusí dávat smysl, ale stroj je zde schopen zachytit kontext.

Jelikož jsme měli takto uměle vytvořený text, bylo náročně určit začátek a konec věty. Zpočátku jsme uvažovali nad jednoduchým parserem, který by vzal 10 slov před a 10 slov po nalezené entitě. Také jsme uvažovali o rozdělení do vět podle hloubky zanoření jednotlivých elementů. Nakonec jsme zkoušeli najít nějaký nastroj pro automatické rozdělování textu do vět. Knihovna `Flair` tuto možnost nenabízí, ale knihovna `SpaCy` nabízí několik možností, jak rozdělit text do vět. Nejjednodušším je nastavení stanovených pravidel pro věty, které můžeme definovat. Lepší metoda, která se více hodila na náš problém, byla rozdělování pomocí již natrénovaného modelu. Při této metodě se někdy stávalo, že rozdělila název produktu mezi dvě věty. Proto jsme ještě před rozdělováním do vět přejmenovali celý název produktu do jednoho slova, který jsme po rozdělení do vět opět navrátili do původního znění. Jelikož metoda rozdělování textu do vět poskytovala i oddělování slov, anotování dat jsme přidělovali podle shody slov. Oddělování slov nám také usnadnilo práci při přiřazování ceny. Někdy se totiž stalo, že text ceny na stránce byl `'45.00€'` Jak jsme výše ukazovali, slovo, které jsme hledali, je `'45.00'` a tím pádem bychom nenašli a nepřiřadili anotaci. Samozřejmě po rozdělení slov pomocí modelu dostaneme dvě slova `'45.00'` a `'€'`, kde bychom již našli shodu při prvním slovu. Do datasetu jsme přidávali pouze ty věty, které obsahovaly alespoň jednu anotaci.

Dalším faktem, který byl velmi překvapující, bylo objevení duplicitních vět v datasetu pro filmy. Některé věty se opakovaly i 10krát. Původní počet získaných vět byl 105 214, po odstranění duplicitních vět nám zbylo pouze 59 511. Ty pravděpodobně vznikly opakováním daných částí stránky, která obsahovala například jméno režiséra apod. U datasetu s produkty jsme také zaznamenali duplicity, avšak ne v tak výrazném rozsahu.

Datasety pro NER používají několik typů, jak definovat a označovat jednotlivé anotace. Úplně základní je BIO a pak se používají také rozšíření jako BIOE a BIOES. Rozdíl mezi jednotlivými typy je ,že rozlišují a dávají více informací modelu. My budeme v naší práci používat tu nejvíce rozšířenou, tedy BIOES která používá následující značky:

- **B (begin):** označuje začátek slova anotace
- **I (inside):** označuje, že je slovo uvnitř anotace
- **O (out):** označuje, že slovo nepatří do žádné anotace
- **E (end):** označuje konec slova anotace
- **S (single):** označuje, že slovo patří do anotace a sestává pouze z jednoho slova

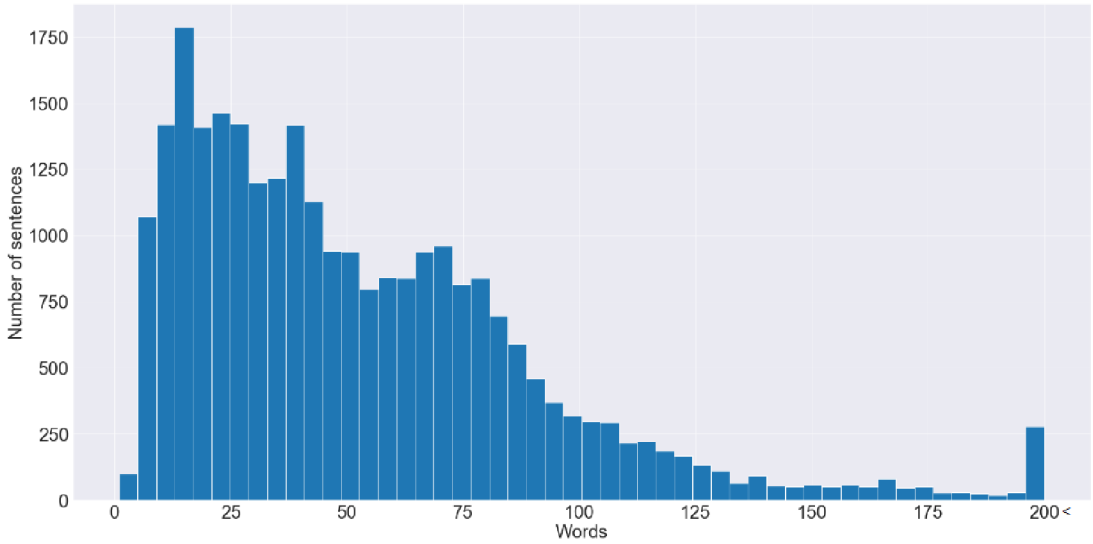
5.2.1 Dataset pro produkty

Dataset pro produkty sestává z deseti různých stránek. Bylo staženo 9 390 stránek, které obsahovaly informace o produktech. Z větší části to byly produkty z oblasti elektroniky, ale snažili jsme se přidat i produkty, které mají běžnější názvy. Detailnější přehled je zobrazen v tabulce 5.1.

Tabulka 5.1: Přehled zdrojů pro dataset Produktů

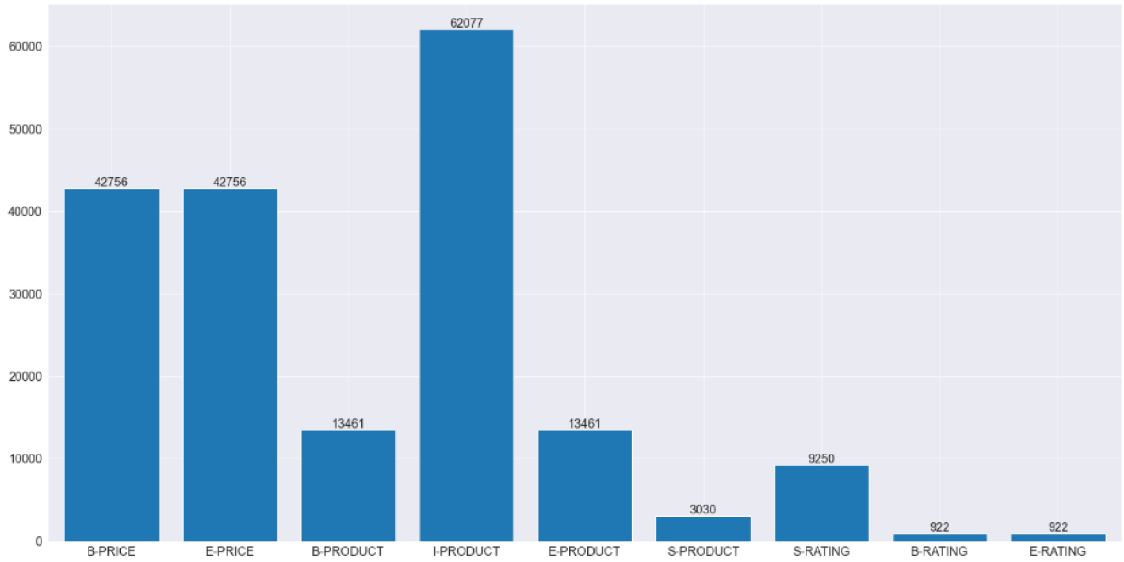
název webů	**počet stažených stránek**
alza.sk/EN	2 723
dell.com	1 039
fbadiscounts.co.uk	691
hp.com	593
jysk.co.uk	2 572
beerrepublic.eu	642
ikea.com	350
northfinder.com	315
versace.com	165
gymbeam.com	300
	9 390

Ze stažených stránek se nám podařilo získat 26 627 anotovaných vět, které obsahují 40 058 unikátních slov. Nejdelší věta se skládá ze 654 slov. Počet vět které sestávají z více než 200 slov je 331. Většinou tyto dlouhé věty sestávají z detailních vlastností produktu a obsahují speciální znaky. Tyto speciální znaky jsou pak použity jako jedno slovo což výrazně prodlužuje délku vět. Četnost slov ve větách je ukázána v následujícím grafu 5.2.



Obrázek 5.2: Četnost slov ve větách pro dataset produktů

Další informace, které jsou popsány na obrázku 5.3, obsahují celkové četnosti jednotlivých tagů.



Obrázek 5.3: Četnost tagů bez tagu O pro dataset produktů

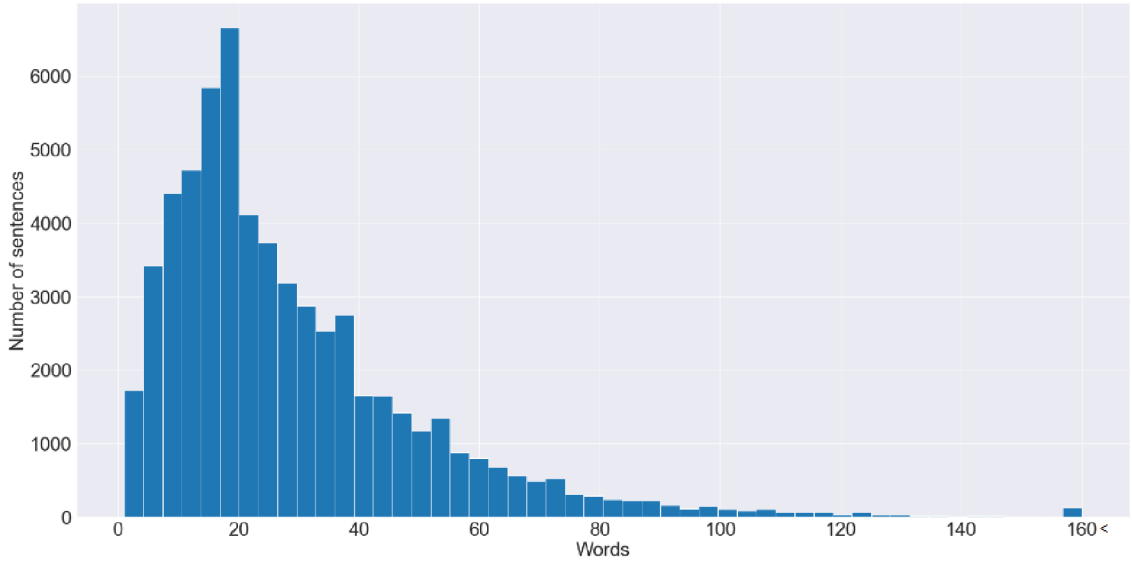
Z tohoto obrázku můžeme vyčíst, že náš dataset obsahuje nejvíce tagů typu **PRICE**, a to celkem 42 756. Tyto tagy se skládají vždy jen ze dvou slov, jelikož nemáme žádné **I-PRICE** tagy. Dalším nejpočetnějším tagem je **PRODUKT**, který se vyskytuje 16 491krát a většinou pozůstává z více než dvou slov. Tento tag se mimo jiné vyskytuje i jako jednoslovný název produktu, a to 3030krát. Nakonec zde máme tag **RATING**, který se vyskytuje 10 172krát v datasetu. Abychom viděli rozdíly v grafu mezi jednotlivými četnostmi, vynechali jsme tag **O**, který je samozřejmě nejpočetnější a vyskytuje se až 1 564 700krát.

5.2.2 Dataset pro filmy

Dataset filmů se skládá ze dvou hlavních stránek, ze kterých jsme čerpali. Sbírali jsme názvy filmů a seriálů. Pro stránku rottentomatoes.com se nám podařilo posbírat 39 318 stránek. Jakmile jsme však přišli na to, že chybí některé části stránek a budeme je otevírat pomocí knihovny `Selenium`, celkový čas výrazně vzrostl, tudíž jsme náhodně vybrali 3 000 linků a následně je zpracovali. Stránky, které neobsahovaly hodnocení filmů, jsme nestahovali. Přehled výsledného počtu stažených stránek je ukázán níže v tabulce 5.2.

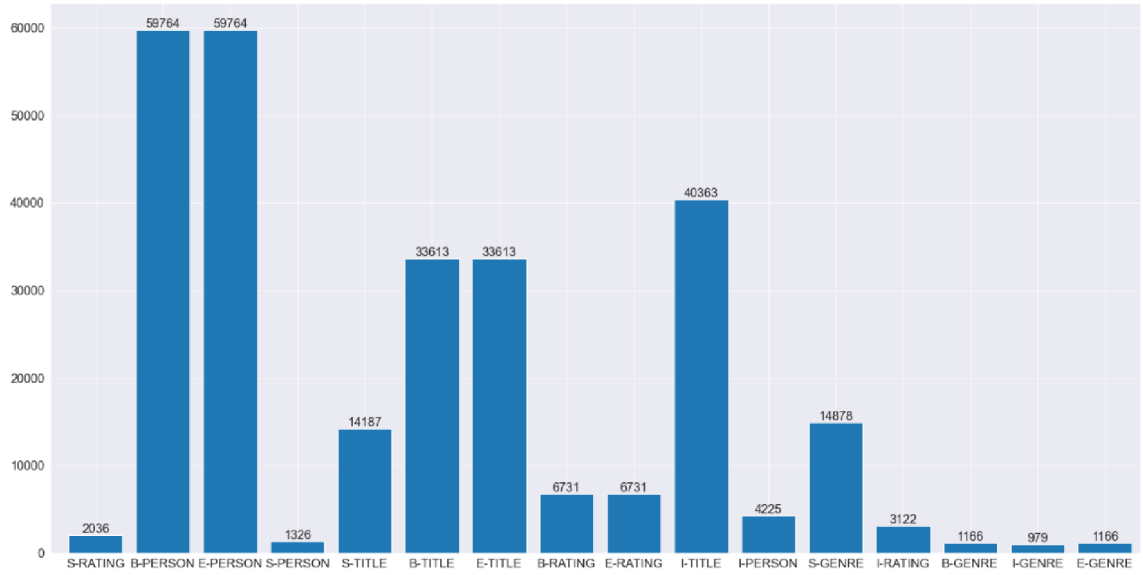
Tabulka 5.2: Přehled zdrojů pro dataset Filmů

název webů	**typ**	**počet stažených stránek**
imdb.com	Filmy	1 989
	Seriály	1 899
rottentomatoes.com	Filmy	1 972
	Seriály	2 250
		8 110



Obrázek 5.4: Četnost slov ve větách pro dataset filmů

Po zpracování stažených stránek a anotování vět jsme získali 59 511 vět, kdy každá věta obsahovala vždy alespoň jednu anotaci. Nejdelší sekvence se skládá ze 458 slov. Počet vět jejichž délka je více než 160 slov je 134 vět. Dataset obsahuje 74 305 originálních slov. Průměrná délka vět je 28.78 slov. Četnosti délky vět můžeme vidět na obrazku 5.4.



Obrázek 5.5: Četnost tagů bez tagu O pro dataset filmů

Jak tedy můžeme vidět na obrazku 5.5, nejpočetnějším tagem je **PERSON**, který se vyskytuje až 61 090krát, což je docela velká četnost vzhledem k tomu, že jsem zpracoval pouze 8 110 stránek. Počet jmen, která se skládají z více než dvou slov, je málo (což je logické). Velmi mě překvapilo, že existují jména pozůstávající pouze z jedním slovem. Po prozkoumání jsem přišel na to, že se jedná spíše o přezdívky slavných herců - např. Zendaya, Bono, Rufus apod. Druhým nejpočetnějším tagem je **TITLE**, který je v datasetu

47 800krát. Obvykle je tvořen z více slov. Těsně po názvu je tag **GENRE** s počtem 16 044krát, také obsahují i některé názvy žánrů, které se skládají z více slov. Na posledním místě je tag **RATING** s počtem 9 853 pro více slov a 14 735krát s jedním slovem.

5.3 Vytvoření modelu

Na základě porovnání a vyhodnocení v předcházejících kapitolách jsem se rozhodli využít knihovnu `Flair` pro její vynikající výsledky v porovnání s ostatními knihovnami. Taktéž knihovna nabízí jednoduché definování modelu, který budeme vytvářet pro detekování pojmenovaných entit nad surovým textem webové stránky. Ještě předtím, než se pustíme do popisu vrstev modelu, popíšeme Embeddngs, které jsou nezbytnou součástí zpracování přirozeného jazyka.

5.3.1 Embeddings

Embeddings vrstva se používá k zakódování slov, tokenů nebo znaků na vektory s nízkodimenzionální reprezentací v prostoru. Tyto vektory reprezentují význam jednotlivých slov nebo tokenů, což modelu umožňuje pracovat s těmito prvky v numerickém formátu.

Konkrétní implementace embeddings vrstvy závisí na použitém embedding modelu, který může být předtrénován na velkém korpusu nebo natrénován pro specifickou úlohu. Tyto vektory jsou pak použity k zakódování vstupních slov nebo tokenů, které jsou pak aktualizovány během trénování modelu. To zajišťuje, že se model učí reprezentovat slova a tokeny v prostoru tak, aby byla zaručena nejlepší výkonnost pro konkrétní úkol. Tento parametr má vysoký vliv na trénování modelu, zvolení správné metody pro naši úlohu může vést k vysoké přesnosti. Proto se budeme snažit experimentovat s tímto parametrem. Seznam embeddings které budeme používat při experimentech:

- **CharacterEmbeddings:** je typ embeddings, kde každé slovo nebo token je reprezentován jako sekvence vektorů znaků. Na rozdíl od tradičních word embeddings, kde každé slovo je reprezentováno jako jeden vektor, v případě CharacterEmbeddings je každé slovo rozděleno na jednotlivé znaky, které jsou pak reprezentovány pomocí samostatných vektorů. Tento přístup umožňuje zachytit větší míru variability v textu, včetně nových slov a slov s nestandartními tvary.

  CharacterEmbeddings mohou být také použity jako doplňková informace k tradičním word embeddings. V tomto případě se vektor znaků spojí s vektorem reprezentujícím slovo a vytvoří se kombinovaný vektor, který se použije jako reprezentace slova. Tento přístup umožňuje zachytit více úrovní významu v textu a může zlepšit výsledky.

- **BPE:** Byte Pair Encoding využívá metodu, kde je každé slovo nebo podslovo reprezentováno vektory. BPE algoritmus se skládá ze dvou kroků. První krok spočívá v tom, že algoritmus postupně nahrazuje dvojice znaků (resp. bytů) v textu novým symbolem, který se ještě v textu nevyskytl. Tento proces se opakuje, dokud se nedosáhne požadovaného počtu tokenů nebo dokud se nedosáhne stanoveného limitu počtu iterací. Výsledkem je slovník nových symbolů a jejich frekvence.

  Druhý krok spočívá v tokenizaci textu. Text se nejprve rozdělí na jednotlivá slova, která se následně tokenizují na podslova. Tokenizace se uskutečňuje tak, že se text rozdělí na jednotlivé znaky, které se následně spárují do dvojic a tyto dvojice se porovnají se slovníkem nových symbolů. Pokud se najde shoda, dané podslovo se

nahradí novým symbolem. Tento proces se opakuje, dokud se nedosáhne posledního symbolu ve slově.

BPE embeddings jsou poté vypočítané jakožto vážený průměr vektorů jednotlivých podslov ve slovu. Tímto způsobem jsou vytvořené embeddings, které jsou odolnější vůči neznámým slovům a méně náchylné na pretrénovaní v porovnání s tradičními word embeddings.

- **GloVe:** Global Vectors for Word Representation je metoda, která využívá globální statistickou informaci o výskytu slov v korpusech. Následně pomocí této metody vytvoří vektorovou reprezentaci slov, která zachycují jejich významovou podobnost a vztahy. Používá metodu vyskytujících se součinů, která se vypočítá z frekvencí výskytu slov v textu. Vypočtené součiny se pak normalizují a transformují do matice, která se pak dekomponuje pomocí SVD[1] na vektorové reprezentace slov. GloVe embeddings jsou používány pro jejich schopnost zachytit jemné sémantické a syntaktické vztahy mezi slovy, jako například synonymní a antonymické vztahy, vztahy mezi slovy a jejich kontextem nebo významové relace mezi slovy, jako například `"king"` a `"queen "`. GloVe embeddings jsou jedním z nejpoužívanějších způsobů, jak reprezentovat slova v jazykových modelech a dosáhnout vysokých výsledků na různých jazykových úkolech.

- **BERT:** Bidirectional Encoder Representations from Transformers je model pro predikci kontextu slov, který byl vyvinut Googlem v roce 2018. Je to velká nepřerušovaná neuronová síť, která se trénuje na velkém množství textových dat. BERT využívá architekturu Transformer, která umožňuje modelu učit se kontextu slov v obou směrech (zleva doprava a zprava doleva), což mu umožňuje pochopit jemné nuance významu slov. BERT používá techniku „maskování“ (masking), která umožňuje modelu učit se vztahy mezi slovy v kontextu. Při trénování se určitá slova vstupního textu náhodně zamaskují a model se snaží předpovídat jejich původní hodnotu na základě kontextu kolem nich.

- **Flair:** Použití takzvaných směrových embeddingů se slova v textu zpracovávají ve dvou směrech - od začátku a od konce věty. Forward embedding se tvoří tak, že prochází věty zleva doprava a trénuje vektory slov v této sekvenci. Backward embedding se tvoří tak, že prochází věty zprava doleva a trénuje vektory slov v opačném pořadí. Tímto způsobem se zachytí různé aspekty kontextu, které se projevují v různých částech věty. Například slovo `"bank"` v angličtině může mít různý význam, jelikož se nachází v různých kontextech. Pokud se použijí pouze forward embeddings, možná nezachytíme význam slova `"bank"` jako finanční instituce, nachází-li se na konci věty. V tomto případě použití kombinace forward a backward embeddings umožní zachytit kontextové významy slov v celé větě.

Také knihovna nabízí takzvané **Stacking Embeddings**, což nám umožňuje zkombinovat více typů embedding. Tyto embeddings jsou jednoduše zřetězené. Například máme-li embeddings na úrovni znaků, může být vhodné přidat další embedding, který rozšíří úroveň na slova. V tomto případě by bylo zastoupení následovné:

$$W_i = \begin{bmatrix} W_i^{CharLM} \\ W_i^{CloVe} \end{bmatrix} \tag{5.1}$$

[1] https://www.sciencedirect.com/topics/mathematics/singular-value-decomposition

5.3.2 Popis modelu

Jak jsme již popisovali v kapitole 2.3.3, kde jsme ukázali architekturu modelu, kterou budeme používat BI-LSTM-CRF.

První nastavení modelu sestává ze zvolení embeddings, které chceme použít pro náš dataset. Jednotlivé embeddings, které jsme použili v naší práci, jsme si popsali výše. Potom vstupní slova jsou reprezentovány jako vektory slov. Tyto vektory jsou pak zpracovávány druhou vrstvou, která je tvořena z rekurentních neuronových sítí, kde jsme nastavili velikost skryté vrstvy na 256 pro obě sítě (forward-LSTM, backward-LSTM). Výstup z této vrstvy můžeme definovat jako konketanaci vektorů:

$$r_i = \begin{bmatrix} r_i^f \\ r_i^b \end{bmatrix} \tag{5.2}$$

Kde r_i^f a r_i^b jsou dopředným a zpětným výstupem stavů pro BI-LSTM vrstvu. Pro tuto vrstvu je ještě nastaven WordDropout o hodnotě 5%. Tento parametr vyjadřuje pravděpodobnosti, s jakou bude dané slovo nahrazeno náhodným tokenem. Tímto se snažíme zvýšit výkonnost modelu a zabránit mu, aby se přetrénoval. Další parametr, který nastavujeme, je Locked Dropout na hodnotu 50%, tento parametr slouží ke snížení možnosti přetrénování neuronových sítí. Principem je "vypínání"některých neuronů v každé vrstvě neuronové sítě během učení.

Poslední vrstva sestává z CRF, která vypočítává nejpravděpodobnější posloupnost tagů pro celou vstupní sekvenci. Detailněji to můžeme vyjádřit následovně. Pro vstupní větu $X = (x_1, x_2, ...x_n)$, výstup z BI-LSTM lze jednoduše převést na matici skóre označenou jako P. Tato matice bude velikosti $n \cdot k + 2$, kde k je počet vstupních znaků zvýšen o dvě značky `<start>` a `<end>`. Pak $P_{i,j}$ představuje skóre $j - teho$ tagu $i - teho$ slova ve větě. A pro danou sekvenci predikcí $y = (y_1, y_2, ..., y_n)$ můžeme definovat jejich skóre jako:

$$s(X, y) = \sum_{i=0}^{n} A_{y_i, y_{i+1}} + \sum_{i=1}^{n} A_{P_i, y_i} \tag{5.3}$$

kde A je matice přechodného skóre a $A_{i,j}$ representuje skóre přechodu z tagu i na tag j. Tag y_0 a y_n představují tagy `<start>` a `<end>`.

Softmax nad všemi možnými sekvencemi y vypočítáme jako:

$$p(y|X) = \frac{e^{s(X,y)}}{\sum_{\tilde{y} \in Y_x} e^{s(X,\tilde{y})}} \tag{5.4}$$

Kde Y_x je množina všech možných sekvencí anotací pro větu X. Během trénování se snažíme maximalizovat logaritmickou pravděpodobnost sekvence:

$$log(p(y|X)) = s(X, y) - log\left(\sum_{\tilde{y} \in Y_x} e^{s(X,\tilde{y})}\right) \tag{5.5}$$

V procesu dekódování se algoritmus dynamického programování používá k výpočtu skóre věty, která nakonec vybere sekvenci značení. Flair v tomto případě používá algoritmus Viterbi[2]. Sekvence s nejvyšším skóre může být vyjádřena jako:

[2] https://www.sciencedirect.com/topics/mathematics/viterbi-algorithm

$$y^* = \arg\max_{\tilde{y} \in Y_x} s(X, \tilde{y}). \tag{5.6}$$

Přehled nastavených parametrů pro trénování modelu je v tabulce 5.3. Tyto parametry jsme postupně upravovali pro nejlepší výsledky a pro naše kapacity GPU.

Tabulka 5.3: Přehled nastavení modelu pro trénování

Atribut	hodnota
BiLSTM hidden layer	256
Word droput	0.05
Locked droput	0.5
Learning rate	0.001
Optimizer	Adam
Batch Size	32
Epochs	10
Patience	3

Kapitola 6

Experimenty a vyhodnocení

V této kapitole bude představena metoda vykonávání experimentů s výše definovaným modelem a jeho modifikacemi. Budeme zde porovnávat výsledky trénovaných modelů. Experimenty budeme provádět nad oběma vytvořenými datovými sadami.

6.1 Postup experimentování

Jelikož budeme vyhodnocovat přesnost daných modelů, jako první si definujeme způsob, jakým budeme měřit jejich přesnost.

Vyhodnocení modelů budeme provádět nad umělými větami, které budou obsahovat anotovaná data. Budeme porovnávat přesnost modelů během trénování. To bude zahrnovat porovnávání loss funkce během trénování a výsledky modelů jako f1-skóre, precision a recall pro každý tag zvlášť. Tyto statistické metody se vypočítají jako v rovnicích 4.1, 4.2 a 4.3. Další metrikou je Accuracy, kterou vypočítáme následovně:

$$\text{accuracy} = \frac{\text{Počet správně klasifikovaných}}{\text{Celkový počet příkladů}} \tag{6.1}$$

Tímto experimentem bude srovnání výsledků přesností modelů pro různé druhy embeddings. Tyto Druhy embeddings a jejich význam jsme popisovali detailněji již v kapitole 5.3.1, kde jsme si představili všechny druhy, které zahrneme v našem experimentu. Jednoduchý přehled embeddings, které nabízí knihovna flair a se kterými budeme pracovat, je popisován v tabulce 6.1.

Tabulka 6.1: Přehled embeddings

Název	typ	značka
GloVe	statický word embedding	GloVe
Flair(forward,beckward)	kontextualizovaný word embeddings	Flair
CharacterEmbeddings	embeddings na úrovni znaků	Char
BPE	byte-pair embeddings	Bpe
Bert Transformer	kontextualizovaný transformer	Bert

Tabulka níže ukazuje, s jakým nastavením vektorových reprezentací jsem experimentoval. Pro oba datasety jsem zkoušel stejná nastavení embeddings.

Tabulka 6.2: Přehled embeddings

konfigurace
GloVe
Flair
Bert
Bpe + Char
GloVe + Flair
Bert + Flair

Abychom dosáhli porovnatelnějších výsledků, každý model používal stejný validační soubor, kde jsem používal strategii 80% trénovací, 10% validační, 10% testovací. Tento experiment je rozdělen na dva podexperimenty, protože jeden bude prováděn pro dataset produktů a další pro dataset filmů.

6.1.1 Experiment nad datasetem produktů

Protože dataset produktů sestává z více stránek, kde některé mají větší zastoupení než jiné, validační soubor sestává z 10% datasetu každé stránky. Toto rozdělení jsem provedl i proto, že některé stránky obsahují elektroniku, ale jiné zase nábytek, což je náročnější na detekci. Ideální by bylo, kdyby každá strana měla stejné zastoupení ve validačním souboru. To však není v našem případě možné, protože pro některé specifické oblasti máme málo dat. Například bychom zkoušeli testovat detekci názvů produktů nábytku, ale v našem datasetu bychom vůbec neměli žádná data s jejich názvy, ale pouze s názvy elektroniky. Výsledky by byly zkreslené, proto jsme to řešili tímto způsobem.

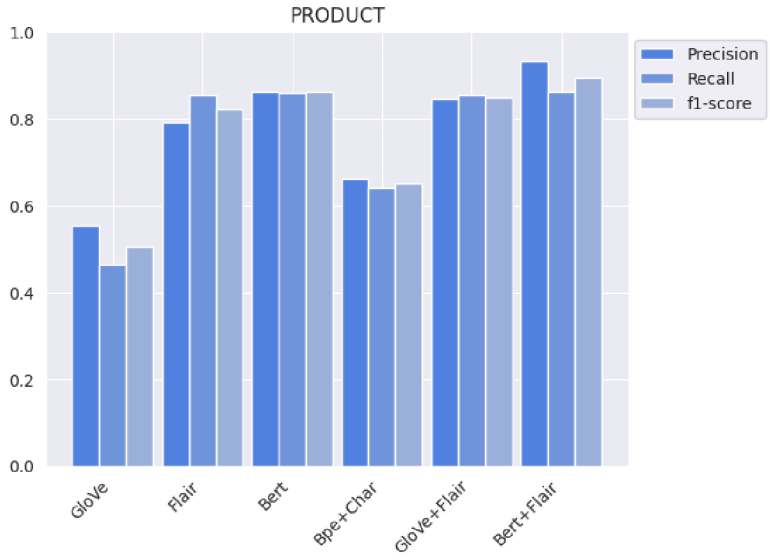
Níže můžeme vidět dosažené výsledky modelů pro dané konfigurace embeddings.

Tabulka 6.3: Přehled výsledků pro dataset Produktů

Embeddings	**CFR vrstva**	**Loss function**	**Přesnost**	**F(micro)**	**F(macro)**
GloVe	ano	Viterbi	0.7344	0.8469	0.7518
Flair	ano	Viterbi	0.8698	0.9304	0.8860
Bert	ne	Cross Entropy	0.8814	0.9369	0.8808
Bpe + Char	ano	Viterbi	0.7865	8805	0.8075
GloVe + Flair	ano	Viterbi	0.8829	0.9378	0.8928
Bert + Flair	ne	Cross Entropy	0.9117	0.9538	0.9143

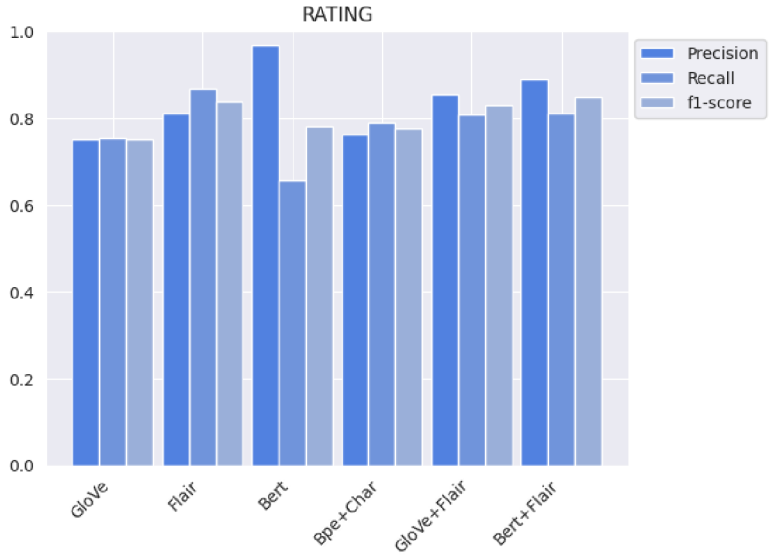
CRF vrstvu jsme museli odstranit z modelu, když jsem použil transformer Bert. Hlavním důvodem bylo, že vrstva CRF nebyla kompatibilní s architekturou Bert.

Dále si ukážeme, jak dopadly přesnosti jednotlivých značek na grafech 6.1, 6.2 a 6.3.



Obrázek 6.1: Výsledky detekce tagu PRODUCT

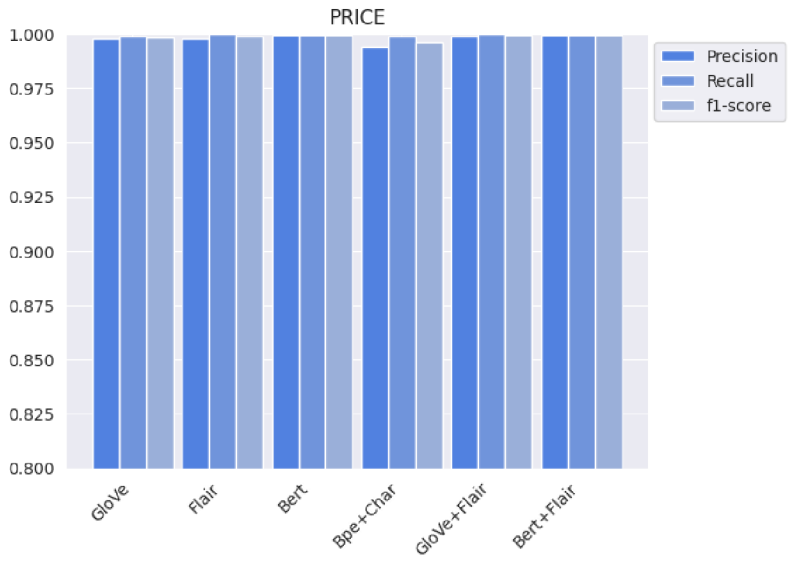
Jak můžeme vidět, nejlepších výsledků dosahuje model s embeddings Bert+Flair, kterému se úspěšně daří detekovat skutečné označené entity. Těsně za tímto modelem jsou modely s embeddings GloVe+Flair a Bert, které mají nižší precision. Následuje model s embeddings Flair, který má skoro stejnou citlivost vůči předchozím dvěma modelům. Bohužel již není tak spolehlivý při určování skutečně označených entit. Poslední dva modely s embeddings Glove a Bpe + Char jsou docela slabé a nedosahují dostatečné přesnosti.



Obrázek 6.2: Výsledky detekce tagu RATING

Model s embeddings Bert má zajímavé výsledky, kde se s detekcí realně označených dat přiblížil nad 90 procent. Na druhou stranu má nejnižší Recall, z čehož můžeme usoudit, že model je selektivní a klasifikuje pouze ty instance, o kterých je si relativně jistý, že jsou pozitivní.To znamená, že model dává přednost minimalizaci falešně pozitivních případů

(false positives) na úkor vyhledávání všech skutečně pozitivních případů. To může být způsobeno nepřesnostmi datové sady, ve které se vyskytují neoznačené entity. Například, že ze stránky dokážeme získáť pouze jednu hodnotu ratingu, přestože stránka může obsahovat i více hodnot ratingu k jiným produktům.



Obrázek 6.3: Výsledky detekce tagu PRICE

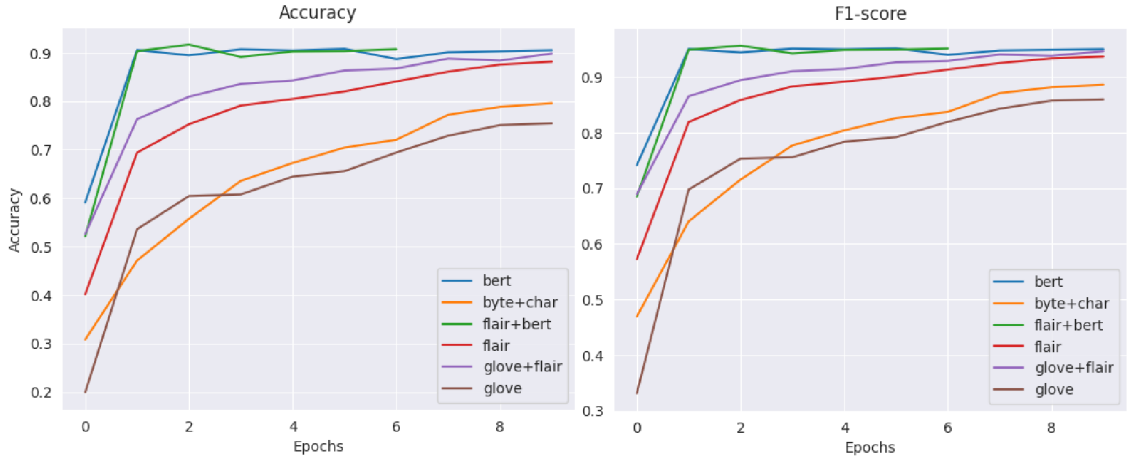
Za zajímavý výsledek můžeme pokládat dosažení velmi vysoké přesnosti pro každý typ embeddings. Je to způsobeno tím, že cenu je možné velmi jednoduše detekovat, tedy alespoň v našem datasetu, protože vždy je to kombinací znaku, měny a částky. Jelikož se na stránkách s produktem moc nevyskytuje cena ve slovním podání, je to poměrně jednodušší.

Přesné hodnoty přesnosti detekce tagů je popsána v tabulce níže 6.4.

Tabulka 6.4: Přesnosti pro jednotlivé tagy pro různé embeddings

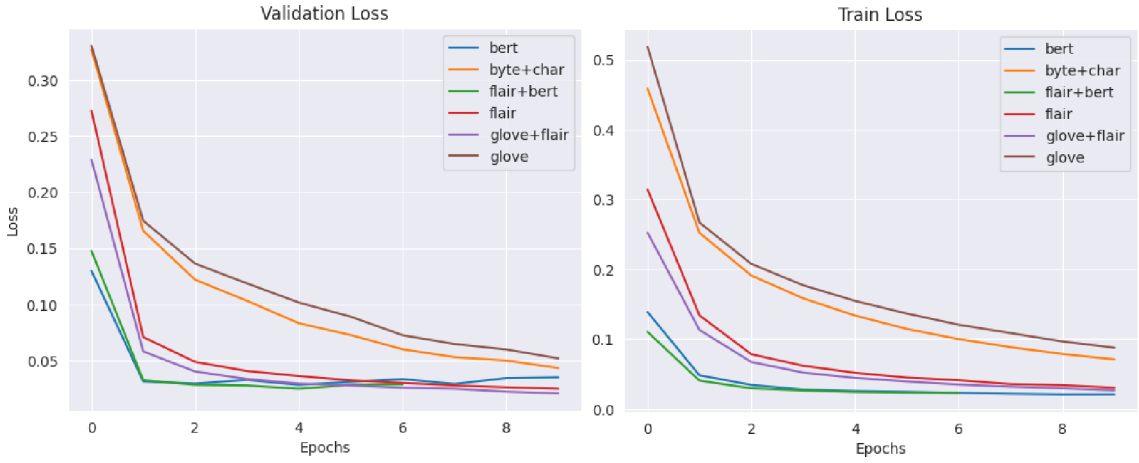
		Glove	Flair	Bert	Bpe+Char	Glove+Flair	Bert+Flair
Product	prec.	0.5548	0.7922	0.8627	0.6628	0.8450	0.9318
	recall	0.4633	0.8540	0.8593	0.6393	0.8527	0.8621
	F1	0.5049	0.8219	0.8610	0.6508	0.8489	0.8956
Price	prec.	0.9976	0.9976	0.9995	0.9942	0.9989	0.9995
	recall	0.9989	1.0000	0.9995	0.9987	1.0000	0.9997
	F1	0.9982	0.9988	0.9995	0.9964	0.9995	0.9996
Rating	prec.	0.7503	0.8104	0.9673	0.7616	0.8536	0.8884
	recall	0.7543	0.8660	0.6561	0.7898	0.8077	0.8105
	F1	0.7523	0.8373	0.7819	0.7754	0.8300	0.8477

Na dalších grafech 6.4 a 6.5 si ukážeme průběh trénování a zlepšování modelů.



Obrázek 6.4: Průběh trénování accuracy a F1-score pro dataset produktů

Na tomto grafu můžeme vidět průběh trénování a zlepšování modelů. Modely, které měly Bert embeddings, se natrénovaly velmi rychle a poté jejich přesnost začala oscilovat na stejné hladině. Jelikož máme nastavený atribut `patience=3`, model flair+bert skončil na sedmé epoše. Model Bert dosáhl ještě zlepšení, a proto nebyl ukončen předčasně.



Obrázek 6.5: Loss nad validačními a trénovacími daty pro dataset produktů

Na validačním grafu můžeme vidět, že model flair+bert se příliš nezlepšuje a začíná se mírně přetrénovávat. Naštěstí byly ukončeny epochy, které zabránily výraznějšímu přetrénování. Model bert však nebyl ukončen a je patrné, že se přetrénoval, což se projevuje ztrátou jeho schopnosti správně generalizovat. Přetrénování začalo prakticky po třetí etapě. Před třetí etapou měl model přesnost 0.9055 na validačních datech.

6.1.2 Experiment nad datasetem filmů

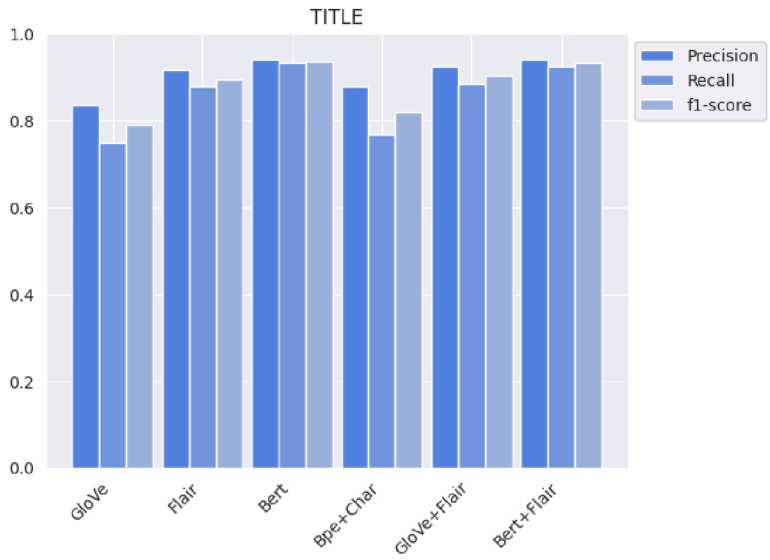
Dataset pro filmy sestává pouze ze dvou stránek. Navíc tyto stránky mají přibližně stejné zastoupení, takže jsme vybrali 10% náhodných vět. Tento validační soubor bude pro všechny modely stejný. Tabulka 6.5 obsahuje základní přehled výsledků trénování.

Tabulka 6.5: Přehled výsledků pro dataset Filmů

Embeddings	**CFR vrstva**	**Loss function**	**Přesnost**	**F(micro)**	**F(macro)**
GloVe	ano	Viterbi	0.6506	0.7878	0.8203
Flair	ano	Viterbi	0.7563	0.8609	0.8705
Bert	ne	Cross Entropy	0.8118	0.8956	0.8926
Bpe + Char	ano	Viterbi	0.6866	0.8136	0.8462
GloVe + Flair	ano	Viterbi	0.7659	0.8672	0.8839
Bert + Flair	ne	Cross Entropy	0.8113	0.8954	0.8992

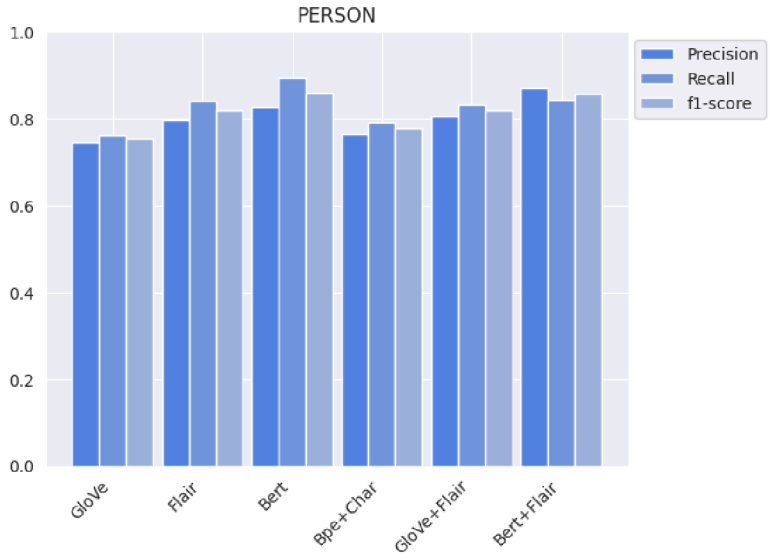
Podobně jako při experimentování s datasetem produktů, i zde jsme museli odstranit CRF vrstvu z modelu u embeddings Bert. Jedná se o velmi zajímavý výsledek pro modely s embeddings Bert a Bert+Flair, které dosahují skoro stejných výsledků. Zde vidíme, že v tomto případě spojení Bert+Flair nepřineslo zlepšení jako u datasetu produktů.

Následně ukážeme v obrázcích 6.6, 6.7, 6.8 a 6.9 jednotlivé výsledky přesnosti tagů.



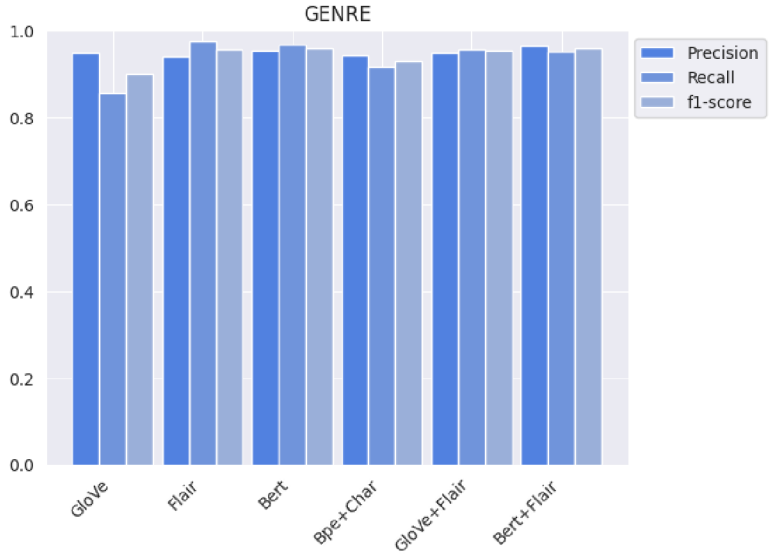
Obrázek 6.6: Výsledky detekce tagu TITLE

Na grafu můžeme vidět, že Bert a Bert+Flair dosahují velmi dobrých výsledků, které mají poměrně stejné hodnoty pro Precision a Recall, což je dobře. Těsně za nimi je model Flair a Flair+Glove.



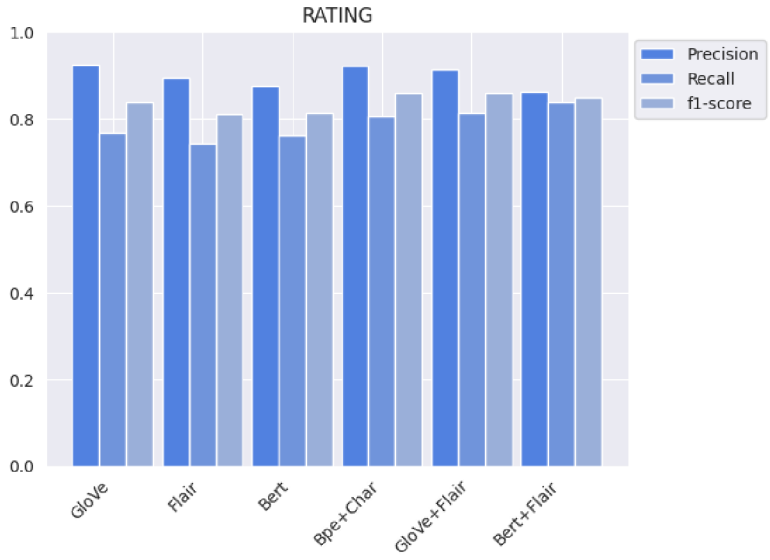
Obrázek 6.7: Výsledky detekce tagu PERSON

Bert dosahoval nejlepší přesnosti pro recall při detekci tagů typu PERSON. Tento model je citlivý a snaží se zachytit co nejvíce skutečně pozitivních případů, i za cenu zvýšeného počtu falešně pozitivních případů. To znamená, že model může klasifikovat některé instance jako pozitivní, i když jsou ve skutečnosti negativní.



Obrázek 6.8: Výsledky detekce tagu GENRE

Všechny modely pro detekci tagu GENRE dosahovaly dobrých výsledků. Model Glove však zaostával a byl zaujatý v detekci tagů. To vedlo k tomu, že označoval i slova, která neměla být označena.



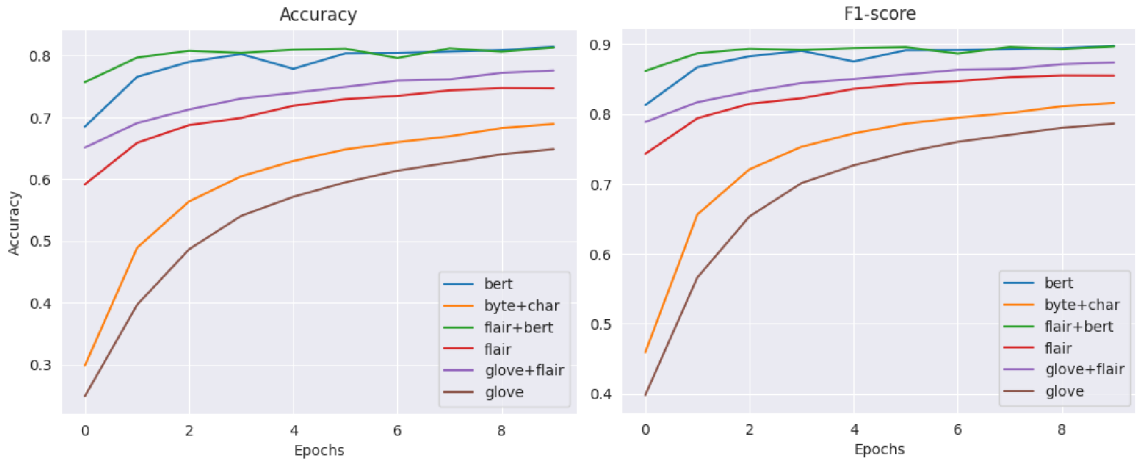
Obrázek 6.9: Výsledky detekce tagu RATING

Stejně tak jako u datasetu produktů, i zde vidíme poměrně větší rozdíly mezi Precision a recall. Tento fakt mohl být způsoben tím, že v datasetu nejsou všechny ratingy označeny. Označeny jsou pouze ty, které jsme získali z JSON-LD, ale pokud tam byl link na jiný film spolu s ratingem, tak jsme ho neoznačili.

Detailní přehled pro přesnější porovnání výsledků pro jednotlivé tagy je zobrazen v tabulce výše 6.6.

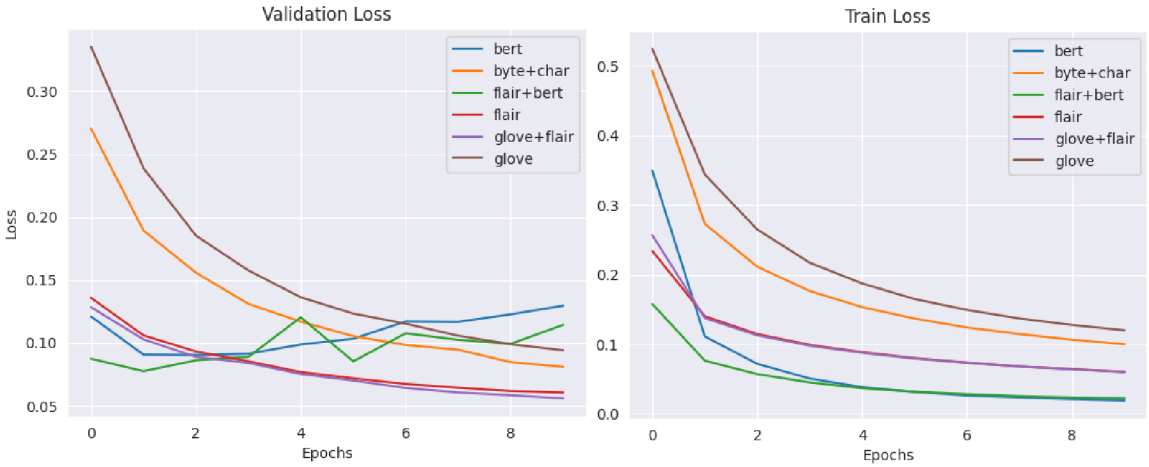
Tabulka 6.6: Přesnosti pro jednotlivé tagy pro různé embeddings

		Glove	Flair	Bert	Bpe+Char	Glove+Flair	Bert+Flair
Title	prec.	0.8345	0.9148	0.9417	0.8772	0.9237	0.9406
	recall	0.7487	0.8772	0.9311	0.7664	0.8845	0.9233
	F1	0.7893	0.8956	0.9363	0.8181	0.9037	0.9319
Person	prec.	0.7462	0.7959	0.8273	0.7632	0.8048	0.8695
	recall	0.7618	0.8399	0.8953	0.7911	0.8334	0.8440
	F1	0.7539	0.8173	0.8600	0.7769	0.8188	0.8565
Genre	prec.	0.9478	0.9413	0.9534	0.9427	0.9494	0.9652
	recall	0.8555	0.9751	0.9665	0.9170	0.9579	0.9524
	F1	0.8993	0.9579	0.9599	0.9297	0.9537	0.9587
Rating	prec.	0.9237	0.8932	0.8750	0.9226	0.9128	0.8628
	recall	0.7682	0.7428	0.7614	0.8059	0.8120	0.8368
	F1	0.8388	0.8111	0.8142	0.8603	0.8595	0.8496



Obrázek 6.10: Průběh trénování accuracy a F1-score pro dataset filmů

Podobně jako u datasetu produktů, i zde vidíme, že oba modely, které používají bert embeddings, začínají oscilovat a dosáhly svého maxima.



Obrázek 6.11: Loss nad validačními a trénovacími daty pro dataset filmů

Jak můžeme vidět, modely, které začaly oscilovat a v podstatě dosáhly svého možného maxima, se začaly přetrénovávat. Oba modely bert a bert+flair se začaly přetrénovávat už po druhé epoše. Model bert dosahuje ve druhé epoše 0.7656 accuracy a model bert+flair 0.7968 accuracy.

6.2 Vyhodnocení experimentu

V tomto experimentu, kde jsme porovnávali výsledky modelů pro různé druhy embeddings, se nám je podařilo natrénovat s celem solidní přesností. Bohužel se některé modely přetrénovaly a jejich přesnost byla proto zkreslená. Toto se nám stalo hlavně pro modely, které používaly transformer Bert. Domnívám se, že jedním z hlavních důvodů, proč se tyto modely tak rychle přetrénovaly, byla jednoduchost datasetů. V podstatě když čerpáme data ze stejné stránky pro různé druhy produktů nebo filmů, struktura stránky se až tak zásadně nemění. Mění se název produktů/filmů, ale náš umělý kontext zůstává stejný. Proto je velmi

důležité zajistit, abychom měli mnoho různých stránek s různou strukturou. V tomto případě jiná struktura stránky znaména jiný kontext, ve kterém zachycujeme entity. Můžeme vidět, že u datasetu produktů, který sestává až z deseti různých stránek, přetrénování daných modelů přišlo mnohem později a nebylo až tak výrazné. Zatímco u datasetu filmů, kde byly jen dvě stranky, přetrénování přišlo hned po druhé epoše.

Myslím, že co se týče kvality datasetu, je stále co zlepšovat, protože byly anotovány automaticky. Tímto způsobem se objevily nežádoucí účinky a vznikly anotace, které byly nesprávně určeny. Ale na opravu těchto nespravných anotací vyplývajících z kontextu bychom museli procházet celý dataset ručně. Toto by bylo velmi časově náročné.

Tímto shrnutím experimentů a prokázanými výsledky bych určitě potvrdil, že rozpoznávání entit nad textem z webové stránky je možné. V podstatě jsme schopni vytvořit umělý kontext, kde osamocená slova spojíme do umělé věty. Tato věta sice nemusí dávat smysl, ale pro detekci dané entity vytváří potřebný okolní kontext. Je také důležité zdůraznit, že stejně jako při trénování rozpoznávání entit v souvislém textu se používá rozsáhlý trénovací dataset, je zde také potřeba velkého množství dat. Alternativou je zvýšit přesnost výsledků modelů pomocí využití embeddings, které byly trénovány na souvislém textu.

Kapitola 7

Závěr

V rámci této práce bylo nutné prostudovat stávající knihovny pro rozpoznávání pojmenovaných entit, a to v programovacím jazyce Python. Na začátku jsme zkoumali metody pro klasifikaci textu v obecném smyslu. Poté jsme se zabývali architekturami, které tyto knihovny využívají, a jak dobře jsou schopny klasifikovat. Po srovnání výsledků jsme se rozhodli pro knihovnu Flair s architekturou BiLSTM-CRF.

Dále jsme zkoumali dostupnost anotačních dat z webu a studovali možnosti sémantického webu, který využívá extrakční jazyk RDF a jeho přítomnost přímo v HTML kódu. Prostudovali jsme také znalostní báze obsahující velké množství anotovaných dat.

Na základě předchozí studie jsme se rozhodli sbírat anotace z JSON-LD definic přímo ze stránek. Definovali jsme anotace, které budou obsahovat naše datasety a vybrali vhodný typ datasetu pro rozpoznávání pojmenovaných entit, konkrétně BIOES formát.

V implementační části jsme stahovali potřebné stránky a prováděli předzpracování a čištění dat. Definovali jsme model a nastavili vhodné hyperparametry.

Dále jsme experimentovali s různými způsoby vektorizace textu a porovnávali výsledky, kdy jsme poté zhodnotili natrénované modely. Na základě těchto testování jsme dospěli k závěru, že je možné rozpoznávat pojmenované entity v textu webových stránek, přičemž vytváření umělých vět nám umožňuje vytvořit dostatečný kontext pro detekci těchto entit. Je však potřeba poznamenat, že samotné výsledky modelů jsou z důvodu nepřesností v datech zkreslené.

Pro další zkoumání této problematiky by bylo vhodné rozšířit počet stránek v jednotlivých datasetech, abychom mohli zachytit různé druhy kontextu. Také bychom museli opravit chyby vznikající v důsledku kontextu. Tímto způsobem bychom získali kvalitnější dataset.

Jednou z možností, která by stála za zvážení, je převzít model, který byl natrénován na souvislém textu, a dotrénovat ho na našem datasetu. Tímto postupem bychom mohli dosáhnout zlepšení výkonu a schopnosti modelu rozpoznávat pojmenované entity na konkrétních webových stránkách.

Na závěr bych tedy zhodnotil, že tento přístup extrakce informací z webu má značný potenciál a nabízí nové možnosti. Je nepochybně možné natrénovat modely, které by byly schopny efektivně detekovat a rozpoznávat entity v textu z webových stránek. Tímto bychom otevřeli dveře k novým perspektivám v oblasti získávání a využití strukturovaných informací na internetu.

Literatura

[1] Auer, S., Bizer, C., Kobilarov, G., Lehmann, J., Cyganiak, R. et al. Dbpedia: A nucleus for a web of open data. In: *The semantic web*. Springer, 2007, s. 722–735.

[2] Brad. *Privacy protection with AI: Survey of data-anonymization techniques* [online]. 2020 [cit. 2023-19-01]. Dostupné z: https://bradpayne.ca/privacy-protection-with-ai-survey-of-data-anonymization-techniques.

[3] Decker, S., Mitra, P. a Melnik, S. Framework for the semantic Web: an RDF tutorial. *IEEE Internet Computing*. 2000, sv. 4, č. 6, s. 68–73. DOI: 10.1109/4236.895018.

[4] Dočekal, M. *Porovnání klasifikačních metod*. Brno, CZ, 2009. Diplomová práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Dostupné z: https://www.fit.vut.cz/study/thesis/22036/.

[5] Central, G. S. *Movie Structured Data* [online]. 2023 [cit. 2023-29-04]. Dostupné z: https://developers.google.com/search/docs/appearance/structured-data/movie.

[6] Central, G. S. *Product Structured Data* [online]. 2023 [cit. 2023-29-04]. Dostupné z: https://developers.google.com/search/docs/appearance/structured-data/product.

[7] Huang, Z., Xu, W. a Yu, K. Bidirectional LSTM-CRF models for sequence tagging. *ArXiv preprint arXiv:1508.01991*. 2015.

[8] Konkol, M. a Konopík, M. Maximum Entropy Named Entity Recognition for Czech Language. In: Habernal, I. a Matoušek, V., ed. *Text, Speech and Dialogue*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, s. 203–210.

[9] Kowsari, K., Jafari Meimandi, K., Heidarysafa, M., Mendu, S., Barnes, L. et al. Text Classification Algorithms: A Survey. *Information*. 2019, sv. 10, s. 1572–9338. DOI: 10.3390/info10040150. ISSN 2078-2489. Dostupné z: https://www.mdpi.com/2078-2489/10/4/150.

[10] Lample, G., Ballesteros, M., Subramanian, S., Kawakami, K. a Dyer, C. Neural architectures for named entity recognition. *ArXiv preprint arXiv:1603.01360*. 2016.

[11] Liu, Z., Lv, X., Liu, K. a Shi, S. Study on SVM Compared with the other Text Classification Methods. In: *2010 Second International Workshop on Education Technology and Computer Science*. 2010, sv. 1, s. 219–222. DOI: 10.1109/ETCS.2010.248.

[12] Mansouri, A., Affendey, L. S. a Mamat, A. Named entity recognition approaches. *International Journal of Computer Science and Network Security*. Citeseer. 2008, sv. 8, č. 2, s. 339–344.

[13] NLTK. *Nltk.org - Natural Language Toolkit* [online]. 2023 [cit. 2023-19-01]. Dostupné z: https://www.nltk.org.

[14] Pellissier Tanon, T., Vrandečić, D., Schaffert, S., Steiner, T. a Pintscher, L. From freebase to wikidata: The great migration. In: *Proceedings of the 25th international conference on world wide web*. 2016, s. 1419–1428.

[15] Piccialli, V. a Sciandrone, M. Nonlinear optimization and support vector machines. *Annals of Operations Research*. 2022, sv. 314, s. 1572–9338. DOI: 10.1007/s10479-022-04655-x. ISSN 1877-7058. CEIS 2011. Dostupné z: https://doi.org/10.1007/s10479-022-04655-x.

[16] W3C. *Resource Description Framework (RDF) Model and Syntax Specification* [online]. 1999 [cit. 2023-20-01]. Dostupné z: https://www.w3.org/TR/1999/REC-rdf-syntax-19990222/.

[17] Serrà, J. a Karatzoglou, A. Getting deep recommenders fit: Bloom embeddings for sparse binary input/output networks. In: *Proceedings of the Eleventh ACM Conference on Recommender Systems*. 2017, s. 279–287.

[18] Foundation, M. *Using shadow DOM* [online]. 2023 [cit. 2023-15-05]. Dostupné z: https://developer.mozilla.org/en-US/docs/Web/API/Web_components/Using_shadow_DOM.

[19] Synek, R. *Klasifikace textu pomocí metody SVM*. Brno, CZ, 2010. Diplomová práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Dostupné z: https://dspace.vutbr.cz/handle/11012/54372.

[20] Techopedia. *Semantic Web* [online]. 2017 [cit. 2023-19-01]. Dostupné z: https://www.techopedia.com/definition/27961/semantic-web.

[21] Vychegzhanin, S. a Kotelnikov, E. Comparison of Named Entity Recognition Tools Applied to News Articles. In: *2019 Ivannikov Ispras Open Conference (ISPRAS)*. 2019, s. 72–77. DOI: 10.1109/ISPRAS47671.2019.00017.

[22] Vychegzhanin, S. a Kotelnikov, E. Comparison of Named Entity Recognition Tools Applied to News Articles. In:. Prosinec 2019, s. 72–77. DOI: 10.1109/ISPRAS47671.2019.00017.

[23] Zhang, W. a Gao, F. An Improvement to Naive Bayes for Text Classification. *Procedia Engineering*. 2011, sv. 15, s. 2160–2164. DOI: https://doi.org/10.1016/j.proeng.2011.08.404. ISSN 1877-7058. CEIS 2011. Dostupné z: https://www.sciencedirect.com/science/article/pii/S1877705811019059.

Příloha A

Obsah paměťového média

- `\doc` PDF s technickou zprávou a LaTeX
- `\datasets` oba datasety pro filmy a produkty
- `\models` obsahují všechny modely z experimentů spolu s výsledky.