STŘEDOŠKOLSKÁ ODBORNÁ ČINNOST

Obor č. 18: Informatika

Vektorová reprezentace textu pomocí grafových neuronových sítí a word2vec váženého typem slov

Theodor Ladin
Praha

Praha 2025

STŘEDOŠKOLSKÁ ODBORNÁ ČINNOST

Obor č. 18: Informatika

Vektorová reprezentace textu pomocí grafových neuronových sítí a word2vec váženého typem slov

Textual embeddings with graph neural networks word-type-weighted word2vec

Autoři: Theodor Ladin
Škola: Gymnázium Praha 7 Nad Štolou 1, Nad Štolou 1/1510, 170 00 Praha 7, Letná
Kraj: Hlavní město Praha
Konzultant: Ing. Lukáš Korel a prof. Ing. RNDr. Martin Holeňa, CSc.

Praha 2025

Prohlášení

Prohlašuji, že jsem svou práci SOČ vypracoval/a samostatně a použil/a jsem pouze prameny a literaturu uvedené v seznamu bibliografických záznamů.

Prohlašuji, že tištěná verze a elektronická verze soutěžní práce SOČ jsou shodné.

Nemám závažný důvod proti zpřístupňování této práce v souladu se zákonem č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) ve znění pozdějších předpisů.

V Praze dne 26.3. Theodor Ladin

Poděkování

Vážený pane inženýre Korele, vážený pane profesore Holeňo, vážení členové mé rodiny, rád bych Vám touto cestou vyjádřil upřímné poděkování za Vaši podporu, pomoc a inspiraci.

Pane inženýre Korele a pane profesore Holeňo, děkuji Vám za cenné rady, odborné vedení a trpělivost, které mi velmi pomohly při mém studiu a práci. Vaše znalosti a zkušenosti pro mě byly neocenitelnou oporou.

Mé velké díky patří také mé rodině, zejména mému bratrovi Mikuláši Ladinovi a mé sestře Anežce Ladinové a mým rodičům za jejich neustálou podporu. Vaše povzbuzení a pomoc mi umožnily překonávat překážky a posouvat se dál.

S vděčností a úctou,

Theodor Ladin

Anotace

Ve své práci SOČ jsem se zabýval způsoby rozpoznání sémantické podobnosti vět. Pokusil jsem se vytvořit efektivní algoritmus, který bude využívat vážení modelu word2vec pomocí slovních druhů. Tento model byl velice efektivní a získal výsledky podobně přesné jako model BERT. Zároveň jsem vytvořil několik modelů grafových neuronových sítí, které dokázaly rozpoznat, zdali dvě věty mají stejný význam. Jeden z modelů dokázal získat podobně kvalitní výsledky jako větší transformery BERT.

Klíčová slova

Grafové neuronové sítě; word2vec; Vektorové reprezentace textu; Grafové konvoluční sítě; Optimalizační metody

Annotation

In my SOČ project, I explored methods for recognizing the semantic similarity of sentences. I attempted to create an efficient algorithm that leverages part-of-speech weighting in the word2vec model. This model proved to be highly effective, achieving results comparable in accuracy to the BERT model. Additionally, I developed several graph neural network models capable of determining whether two sentences have the same meaning. One of these models achieved results of similar quality to larger BERT-based transformers.

Keywords

Graph neural networks; word2vec; Vector representations of text; Graph convolutional networks; Optimization methods

Obsah

1 Úvod

V posledních letech se strojové učení (machine learning, ML) ukázalo jako klíčová technologie s širokým uplatněním napříč mnoha oblastmi. Pokročilé modely strojového učení, známé také jako umělá inteligence (artificial intelligence, AI), se staly základem mnoha inovací. Mezi největší pokroky v ML patří umělé neuronové sítě (artificial neural networks, ANN), které umožňují modelovat složité vzory a vztahy v datech. Největší výzvou je však trénování těchto modelů, zejména rozsáhlých struktur jako jsou transformery, které vyžadují obrovské výpočetní kapacity. Tento problém může být zvlášť relevantní v situacích, kdy je třeba trénovat modely na menších úkolech bez využití náročných výpočetních zdrojů. Cílem této práce je zaměřit se konkrétně na problematiku rozpoznávání významové shody dvou vět formulovaných různými slovy.

Jednou z nejběžněji používaných metod v oblasti zpracování textu pro strojové učení je word2vec. Tato metoda umožňuje efektivní modelování sémantických vztahů mezi slovy pomocí vektorových reprezentací, což vedlo k lepšímu zachycení významových vztahů v rozsáhlých textových korpusech. Díky tomu se word2vec široce používá v aplikacích, jako jsou strojový překlad nebo analýza sentimentu.

Avšak tradiční model word2vec má své limity, zejména pokud jde o zachycení sémantické podobnosti na úrovni celých vět. Na tuto výzvu reagují moderní modely jako Bidirectional Encoder Representations from Transformers (BERT), které rozšiřují možnosti vektorových reprezentací a lépe zohledňují kontext vět. Složité architektury těchto modelů však znamenají vyšší nároky na výpočetní výkon a delší dobu trénování.

Další oblastí, která se v této práci zkoumá, jsou grafové neuronové sítě (graph neural network, GNN). Tyto sítě představují pokročilou metodu hlubokého učení určenou pro efektivní zpracování dat, která lze reprezentovat jako grafy, tedy struktury složené z uzlů a hran. Tento přístup umožňuje modelům lépe zachytit komplexní vztahy mezi entitami, což je užitečné v oblastech jako sociální sítě, biologické systémy nebo znalostní grafy. GNN jsou zvláště efektivní pro porozumění textu, protože umožňují modelovat sémantické vztahy mezi slovy, frázemi nebo větami a zachycovat kontextové závislosti mezi nimi. Na rozdíl od tradičních metod, jako jsou rekurentní neuronové sítě (recurrent neural networks, RNN) nebo transformery, které se zaměřují na sekvenční data, grafové neuronové sítě dokážou lépe zachytit složité struktury a dlouhodobé závislosti.

Cílem tohoto výzkumu je vyvinout efektivní řešení pro problém rozpoznávání sémantické podobnosti mezi větami, které bude schopné fungovat s nižšími nároky na výpočetní výkon. K tomu budou využity nejen jednodušší architektury, ale i složitější přístupy zahrnující grafové neuronové sítě.

Další sekce popisuje využití těchto řešení, teorii za jednotlivými řešeními, předzpracování textu a metodologii jednotlivých řešení. Následující části obsahují výsledky práce a z nich utvořené závěry.

2 POUŽITÉ TECHNOLOGIE

2.1 Knihovny

2.1.1 SpaCy

Jedna z hlavních použitých komponent při zpracování textu je spaCy (1). Má otevřený zdrojový kód a slouží ke zpracování přirozeného jazyka. Tato knihovna určuje slovní druhy, větné členy a rozkládá text na jednotlivé části (tokeny). Datová sada použitých vět, která se používala v tomto úkolu, byla v anglickém jazyce, takže model měl přesné kategorizování, a tudíž byl perfektní tuto úlohu. Model byl využit, kvůli možnosti tokenizace a rozdělení na základní větné členy a slovní druhy. U modelu váženého word2vec (2) byly využity slovní druhy. U grafových neuronových sítí byly využity jak slovní druhy, tak větné členy k přesnému popsání struktury vět.

2.1.2 Pytorch

PyTorch (3) framework má otevřený zdrojový kód, který nabízí flexibilní nástroj pro vytváření a trénování neuronových sítí v jazyce Python (4). Knihovna může být použita na vícevláknové zpracování, definování grafových struktur a využití vektorových matematických operací. Byla použita na definování modelů. Zároveň byly využity základní struktury možné pro grafové neuronové sítě. Protože bylo trénování modelu potřeba maximálně optimalizovat, tak bylo využito různých způsobů dávkového učení a multiprocessingu (5).

2.1.3 Google News Vector Negative 300

Tato komponenta je obecně známá v oblasti zpracování přirozeného jazyka (NLP) (6) a spadá pod označení word2vec. V našem výzkumu byla využita pro transformaci slov do 300dimenzionálních číselných vektorů. Trénována byla na přibližně 1 miliardě slov z Google News článků. Model je vytrénován algoritmem souvislý pytel slov (continuous bag of words, CBOW). Tento model má sémanticky podobná slova umístěna ve vektorovém prostoru blízko k sobě, což bylo využito v našem řešení. Díky obecnosti word2vec modelu jsme mohli výsledky vnoření porovnat v přesnosti s námi vytvořeným řešením i s dalšími modely.

2.1.4 Gensim

Gensim (2) je populární knihovna pro zpracování přirozeného jazyka, která se specializuje na modelování témat a práci s vektorovými reprezentacemi slov. Tato knihovna umožňuje efektivní trénování a využívání modelů jako word2vec. V našem projektu byla použita zejména pro práci s natrénovanými vektorovými reprezentacemi slov a pro výpočet sémantických podobností mezi texty. Díky optimalizovaným algoritmům umožnil Gensim rychlé a efektivní zpracování velkého množství textových dat a přispěl k lepším výsledkům našeho modelu.

2.1.5 Quora Question Pairs

Jako trénovací datová množina pro naše řešení byla použita sada Quora Question Pairs (QQP). Právě možnost využití velkého množství vět v trénovací množině, pomohla v kvalitě přesnosti. QQP obsahuje 400000 vět, kde každá věta je anotována, zdali je sémanticky stejná či rozdílná. Příklady vět z tohoto datasetu jsou vidět tabulce 1.

Tabulka 1

Věta 1	Věta 2	Duplikát
How can I be a good geologist?	What should I do to be a great geologist?	1
What would a Trump presidency mean for current international master's students on an F1 visa?	How will a Trump presidency affect the students presently in US or planning to study in US?	1
How much is 30 kV in HP?	Where can I find a conversion chart for CC to horsepower?	0

2.1.6 Microsoft Research paraphrase

Další korpus, který jsme použili, byl Microsoft Research Paraphrase Corpus (7). Obsahuje přibližně 5800 dvojic vět. Algoritmus jsme trénovali na trénovací části tohoto korpusu, která byla dále rozdělena na validační a trénovací a testovali jej na testovací části. Tento korpus byl primárně využit pro vážený word2vec, protože škála modelu nebyla dostatečně velká, aby data mohl memorizovat (8). To nesdílely grafové neuronové sítě, které měly podstatně větší velikost, takže memorizaci nešlo zabránit, aniž by se data uměla nerozšiřovala.

3 TEORIE

3.1 Vektorové reprezentace textu

3.1.1 Word2Vec

Textové vnoření (Textual Embedding) v NLP je užitečný nástroj. Jedná se o vektorovou reprezentaci textu, která pomáhá zachytit význam vět v mnoharozměrném prostoru (typicky alespoň několik stovek dimenzí). Tato reprezentace je vhodná pro různé úlohy:

3.1.2 Klasifikace textu

- Analýza sentimentu (určuje, zda je věta pozitivní, negativní nebo neutrální)
- Tematická kategorizace (třídí texty do kategorií; např. sport, politika nebo technologie)

3.1.3 Vyhledávání sémantických podobností

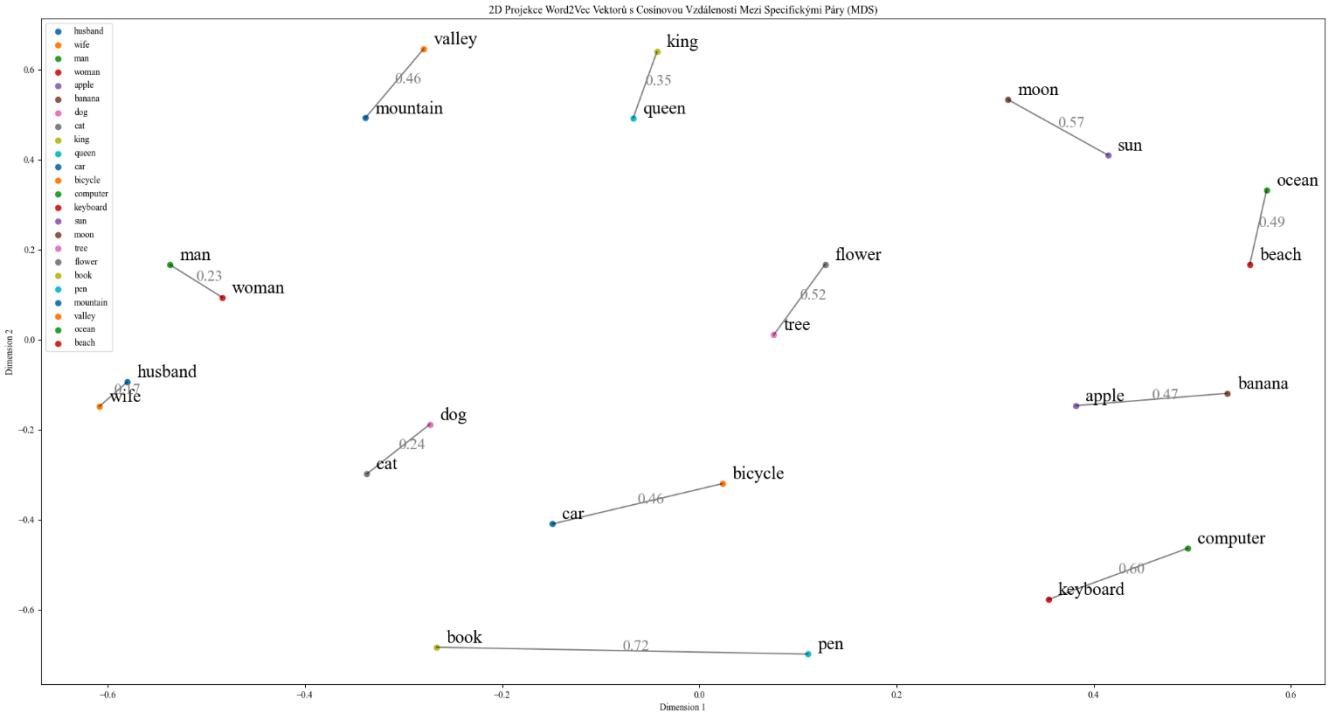
- Detekce parafrází (nachází věty s obdobným významem)
- Informační vyhledávání (vylepšuje výsledky vyhledávání tím, že přesněji přiřazuje dokumenty k dotazům)

3.1.4 Shrnutí textu

- Pomáhá vybírat nejdůležitější části obsahu pro automatické sumarizace
- Celkově usnadňují a zefektivňují práci s textem v široké škále aplikací

Word2vec je jeden z možných způsobů vektorizace (9) slovních tokenů, který využívá vysoké dimenze cílových vektorů na rozlišení významu původních slov. Výhoda tohoto způsobu je, že vektory s podobným významem se nachází v blízkém okolí, tudíž pro měření podobnosti lze využít běžné míry vzdáleností. Oproti eukleidovské vzdálenosti, kde je brána v potaz i vzdálenost bodu v prostoru od průsečíku os, má kosinová vzdálenost (10) tuto vlastnost potlačenou. Kosinová vzdálenost u synonym se blíží k 1 a u antonym k -1. Zároveň je model obousměrný, což nám umožňuje nahlížet na slova pomocí vektorů nebo na vektrory pomocí slov. Je mnoho způsobů, jak vytrénovat takovýto model pomocí tzv. hyperparametrů.

V této práci bylo testováno několik modelů, přičemž konečně byl použit GoogleNewsVector300 – model obsahující 3 miliony 300dimenzionálních vektorů vytrénovaných na článcích Google News. Byl naučen metodou CBOW (11) viz níže. Viz obrázek 1 je projekce tohoto způsobu v 2dimenzionálním prostoru.



Obrázek 1

3.1.5 Souvislý pytel slov

CBOW byl použit na vytrénování modelu Google News Vectors Negative 300 a modelu tohoto výzkumu vytrénovaném na databázi wikipedie. Architektura CBOW využívá okolní slova, aby předpověděla slova následující, a je to jedna z nejčastěji používaných metod v algoritmu word2vec. Metoda je efektivnější pro větší množství dat na trénink. Toto je z důvodu, že se snaží předpovědět jedno slovo z několika kontextových slov, což zjednodušuje výpočty. Nehledí na pořadí slov, v potaz se tedy bere jen jejich samotná přítomnost. CBOW funguje na principu předpovědi cílového slova (target word) na základě okolních slov (context words). Tento model se zaměřuje na výpočet pravděpodobnosti výskytu určitého slova v daném kontextu.

Při trénování modelu CBOW se pro každé cílové slovo vybírá kontextová okna, která obsahují slova před a za tímto cílovým slovem. Model se pokouší na základě těchto okolních slov, která mohou být předchozí i následující) předpovědět cílové slovo. Na rozdíl od modelu Skip-Gram, který se zaměřuje na předpovědi okolních slov na základě cílového, CBOW v podstatě obrací tento proces. Při trénování modelu se tedy používají vstupy – okolní slova – a na jejich základě

se model učí správně odhadnout výstupní slovo (target word). Matematická reprezentace lze vidět v rovnici 1.

Rovnice 1

$$P(w_t \mid w_{t-k}, w_{t-k+1}, \dots, w_{t+k})$$

Matematicky se trénink CBOW modelu zaměřuje na maximalizaci pravděpodobnosti výskytu cílového slova vzhledem k okolním slovům. Pro daný segment textu se snaží minimalizovat ztrátovou funkci, která měří rozdíl mezi skutečným a predikovaným slovem. Typická ztrátová funkce pro CBOW je negativní logaritmus pravděpodobnosti cílového slova. Trénování probíhá optimalizací pomocí metod jako je gradientní sestup, což znamená, že model neustále upravuje své váhy, aby co nejlépe předpověděl slovo v daném kontextu. Rovnice 2 ukazuje matematickou reprezentaci pro negativní logaritmus:

Rovnice 2

$$L = -\log P(w_t \mid \text{context})$$

3.2 Ztrátové funkce

3.2.1 Střední kvadratická chyba

Střední kvadratická chyba (12) měří průměrný kvadratický rozdíl mezi předpokládanou hodnotou a opravdovou hodnotou. Může se tedy používat v našem předpokladu, kde věty můžou být sémanticky opačné, což vede ke kosinové podobnosti menší než nula. Jelikož se vědělo, že kosinová podobnost bude v limitu [-1,1] a předpokládané hodnoty výzkumu v {-1,0,1}, tak hodnota výzkumu bude vždy nenulová pozitivní. Rovnice níže funkci znázorňuje

$$\text{MSE} = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2$$

kde y_i je skutečná hodnota, $\hat{y}_i$ je předpovězená hodnota, a N je počet vzorků. Čím větší je rozdíl $(y_i - \hat{y}_i)$, tím větší je penalizace.

3.2.2 Ztrátová funkce binární křížové entropie

Binární křížová entropie (Binary Cross-Entropy) (13) je ztrátová funkce, která se běžně používá při trénování modelů pro binární klasifikaci. Tento typ úlohy má dvě možné třídy: 0 nebo 1 (např. pozitivní nebo negativní). Binární křížová entropie měří rozdíl mezi skutečnými hodnotami a předpovědí modelu, přičemž penalizuje větší rozdíly. Toto bylo užitečné u standartní grafové konvoluční sítě. Matematicky je definována v rovnici 3.

Rovnice 3

$$\textbf{Binary Cross-Entropy Loss} = -\frac{1}{N}\sum_{i=1}^{N}[y_i \log(\hat{y}_i) + (1 - y_i)\log(1 - \hat{y}_i)]$$

Kde, N je počet vzorků, y_i je skutečná hodnota (0 nebo 1) pro i-tý vzorek, $\hat{y}_i$ je předpovězená pravděpodobnost, že vzorek patří do třídy 1.

Pokud je $y_i = 1$: Tato část vzorce $\log(\hat{y}_i)$ penalizuje, pokud model předpoví nízkou pravděpodobnost pro třídu 1. Čím více se $\hat{y}_i$liší od 1, tím větší je penalizace.

Pokud je $y_i = 0$: Tato část vzorce$(1 - y_i)\log(1 - \hat{y}_i)$ penalizuje, pokud model předpoví vysokou pravděpodobnost pro třídu 1. Čím více se $\hat{y}_i$ liší od 0, tím větší je penalizace.

Křížová entropie tedy měří, jak dobře se model přizpůsobuje skutečným hodnotám. Pokud model předpovídá hodnoty velmi odlišné od skutečných, ztráta bude vysoká, což ukazuje na špatný výkon modelu. Cílem je minimalizovat tuto ztrátu během trénování, aby model co nejlépe aproximoval skutečné třídy.

Tento přístup je vhodný pro úlohy, kde máme dvě třídy, a by se učil vypočítávat pravděpodobnost pro každou třídu (obvykle pomocí sigmoidalní funkce).

3.3 Optimalizační funkce

3.3.1 Quasi-Newton metody

Quasi-Newton metody (14) jsou iterační metody používané na nacházení 0 či lokálního extrému (maxima či minima), pomocí iterační diferenční rovnice. Jsou užitečné pro úlohy, kde je potřeba efektivně najít minimum nebo maximum bez přímého výpočtu druhých derivací. Tyto metody jsou výhodné v případech, kdy výpočet Hessianovy matice by byl příliš nákladný nebo složitý. Iterativní proces v Quasi-Newtonových metodách aktualizuje přiblížení této matice. Místo druhých derivací využívá přibližné hodnoty, které se postupně zlepšují, čímž ušetří čas. Quasi-Newtonovy metody jsou tedy efektivní pro velké optimalizační problémy, jako jsou trénování strojového učení. Jedna z nejvýznamnějších metod je Broyden-Fletcher-Goldfarb-Shanno (15) (BGFS).

3.3.2 Broyden-Fletcher-Goldfarb-Shanno algoritmus

BFGS je jedna z Quasi-Newton metod. Patří tedy mezi iterativní metody, které se používají pro neomezené úlohy nelineární optimalizace. BFGS algoritmus určuje směr sestupu tím, že předzpracovává gradient pomocí informací o zakřivení. Toto dělá zlepšováním předpokladu Hessianovy matice oproti účelové funkci, získané gradientní evaluační metodou sečen. Hlavní cíl je najít minimum funkce $f(x)$ pomocí iterací a změnou proměnných x, kde $x \in R^n$.

Algoritmus funguje tak, že v každém kroku použije invertovanou Hessianovu matici. Metoda mění aproximaci podle gradientu funkce viditeln0m v rovnici 4.

Hlavní aktualizační pravidlo

Rovnice 4

$$x_{k+1} = x_k - \alpha_k H_k \nabla f(x_k)$$

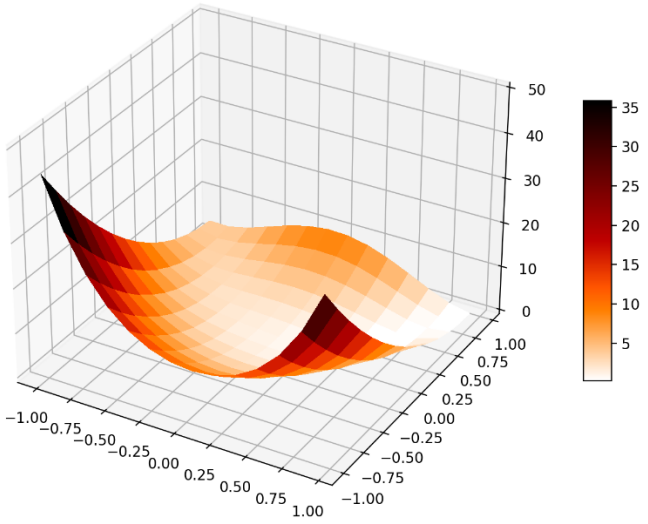
kde x_k je aktuální odhad minima, α_k je krok (nalezený pomocí line search), $\nabla f(x_k)$ je gradient funkce $f(x)$ v bodě x_k, H_k je aproximační matice inverzní Hessovské matice.

Rovnice 5 znázorňuje proces po každé iteraci pro **Aktualizaci matice H_k**

Rovnice 5

$$H_{k+1} = H_k + \frac{y_k y_k^T}{y_k^T s_k} - \frac{H_k s_k s_k^T H_k}{s_k^T H_k s_k}$$

$y_k = \nabla f(x_{k+1}) - \nabla f(x_k)$ je rozdíl gradientů mezi dvěma iteracemi, $s_k = x_{k+1} - x_k$ je změna pozice mezi dvěma iteracemi. Grafické znázornění BFGS metody lze vidět na obrázku 2.



Obrázek 2: Grafické znázornění BFGS metody (16)

3.3.3 Nelder-Mead metoda

Toto je objektivní metoda fungující ve vícedimenzionálním prostoru. Používá se na funkce s větším počtem proměnných, aniž by využila derivací dané funkce. Tudíž je užitečná u funkcích, kde je těžké získat derivaci. Jádrem Nelder-Meadovy (16) metody je koncept simplexu, což je geometrický útvar složený z n+1 vrcholů v n-rozměrném prostoru. Algoritmicky proběhnou tyto čtyři kroky, které evolvují simplex.

Reflexe: Algoritmus odrazí nejhorší vrchol (vrchol s nejvyšší hodnotou účelové funkce) přes těžiště zbývajících vrcholů. Cílem je prozkoumat nové oblasti prostoru, kde by funkční hodnota mohla být nižší.

Reflexe se provádí vzorcem v rovnici 6.

Rovnice 6

$$x_r = c + \alpha(c - x_h)$$

kde x_r je nový vrchol (odraz), c je těžiště ostatních vrcholů, x_h je nejhorší vrchol, α je koeficient, který určuje délku odrazu (obvykle $\alpha = 1$).

Expanze: Pokud reflexe vede k výrazně lepší funkční hodnotě, algoritmus rozšiřuje simplex ve stejném směru, aby urychlil konvergenci k minimu. Viz rovnice 7.

Rovnice 7

$$x_e = c + \gamma(c - x_h)$$

kde x_e je nový vrchol po expanzi, γ je koeficient expanze (obvykle $\gamma > 1$)

Kontrakce: Pokud reflexe nepřinese zlepšení, algoritmus zmenší simplex směrem k lepším vrcholům, aby zpřesnil vyhledávání v menší oblasti. Viz rovnice 8.

Rovnice 8

$$x_c = c + \beta(c - x_h)$$

kde x_c je nový vrchol po kontrakci, β je koeficient kontrakce (obvykle $\beta < 1$)

Zmenšení: Pokud ani kontrakce nevede ke zlepšení, algoritmus zmenší celý simplex směrem k nejlepšímu vrcholu. Tím se redukuje vyhledávací prostor a soustředí se na perspektivnější oblast. Viz rovnice 9.

Rovnice 9

$$x_i = x_{\text{best}} + \delta(x_i - x_{\text{best}})$$

kde x_i je nový vrchol po zmenšení, x_{best} je nejlepší vrchol, δ je koeficient zmenšení (obvykle $0 < \delta < 1$)

Nelder-Mead se používá z důvodu své jednoduchosti a efektivnosti.

3.3.4 Bayesovská Optimalizace

Bayesovská optimalizace (17) je metoda optimalizace černých skříněk, kdy se modeluje pravděpodobnostní distribuce funkčních hodnot a vybírá se bod s největší informační hodnotou. Používá se v hyperparametrické optimalizaci, kde se snažíme minimalizovat počet testovaných konfigurací. Například se dá využít u trénování neuronových sítí, které mají problémy s memorizací. Může upravovat hyperparametry jako je dropout (pravděpodobnost, že specifické váhy nebudou využity), weight decay (proměnná, která snižuje, jak vysoko mohou váhy dosáhnout) a learn rate (v optimalizátorech jako je například Adam, určuje rychlost změny vah).

3.3.5 Optimalizátor Adam

Adam (18) je stochastická gradientová optimalizace. Rozdíl od standartních gradientových metod, se tato dá používat ve složitějších sítích, aniž by se objevily problémy s mírou učení. Toto se děje, protože Adam míru učení mění dynamicky podle předešlých gradientů. Kvůli dynamické křivce je konvergence také rychlejší. Matematické reprezentace jednotlivých kroků jsou vidět v rovnicích 10 až 13.

Odhad prvního momentu (klouzavý průměr gradientů):

Rovnice 10

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t$$

Odhad druhého momentu (klouzavý průměr druhých mocnin gradientů):

Rovnice 11

$$v_t = \beta_2 v_{t-1} + (1 - \beta_2) g_t^2$$

Oprava zkreslení:

Rovnice 12

$$\widehat{m_t} = \frac{m_t}{1-\beta_1^t}$$

Pravidlo aktualizace parametrů:

Rovnice 13

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{\widehat{v_t}}+\epsilon} \widehat{m_t}$$

3.4 Evaluační metriky

3.4.1 Accuracy

Accuracy (19) (správnost) je základní metrika pro hodnocení výkonu klasifikačních modelů. Je definována jako podíl správně klasifikovaných příkladů (Pravé pozitivní TP + Pravé negativní TN) na celkovém počtu příkladů. Tím se získá přesný poměr správných výsledků. Vzorec pro výpočet je následující v rovnici 14.

Rovnice 14

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}}$$

Accuracy je často používána v úlohách, kde jsou třídy vyvážené, avšak její nevýhodou je, že může být zavádějící, pokud jsou třídy nevyvážené (např. v případě velmi častých negativních tříd). Z toho důvodu to nebyla jediná použitá metrika.

3.4.2 F1 Score

F1 skóre (19) je harmonický průměr mezi přesností (precision) a odvoláním (recall). Je to důležitá metrika pro případy, kdy jsou nevyvážené třídy. F1 skóre se vypočítá podle následujícího vzorce v rovnici 15.

Rovnice 15

$$F1 = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

F1 skóre je užitečné, pokud je cílem dosáhnout rovnováhy mezi přesností a odvoláním, zejména v případech, kdy jde o nevyvážené třídy.

3.4.3 AUC (Area Under the Curve)

AUC (19) (oblast pod křivkou) se týká ROC (Receiver Operating Characteristic) křivky, která zobrazuje výkon klasifikačního modelu při různých prahových hodnotách. AUC měří schopnost modelu správně rozlišit mezi pozitivními a negativními třídami. AUC může nabývat hodnot od 0 do 1, přičemž:

- AUC = 0.5 znamená, že model se chová jako náhodný (neexistuje žádná schopnost rozlišování),
- AUC = 1 znamená dokonalou klasifikaci.

AUC je obzvlášť užitečné v případech, kdy jsou nevyvážené třídy, protože se nebere v úvahu konkrétní prahová hodnota pro rozhodnutí. Toto je velice užitečné v případě, kdy se hodnoty

rozdělovaly na různých prazích u specifických modelů. Toto je vidět ve výsledcích práce, kde základní word2vec měl definované věty stejné a rozdílné naprosto jinak.

3.4.4 Friedman Test

Friedmanův test (20) je neparametrický test pro analýzu rozdílů mezi více než dvěma skupinami. Je to statistický test, který je alternativou k opakovaným měřením analýzy rozptylu (ANOVA), která vyžaduje normalitu dat. Používá se k porovnání více než dvou podmínek nebo metod, kdy jsou závislé vzorky. Používá se v případech, kdy je několik podmínek na sobě vzájemně závislé, například při testování několika algoritmů na stejných datech. Toto se perfektně hodilo když bylo otestováno několik modelů.

3.4.5 Wilcoxon Signed Rank Test

Wilcoxonův podepsaný test (21) je neparametrický a se používá k porovnání měření nebo dvou souvisejících vzorků. Je to alternativa k párovému t-testu, který předpokládá normalitu dat. Tento test se používá pro situace, kdy jsou dvě související skupiny nebo měření (např. před a po experimentu) a chtějí se testovat, zda existuje statisticky významný rozdíl mezi těmito dvěma měřeními. Toto mohlo být využito na porovnání mezi neváženým a váženým word2vec. Zároveň byla porovnána schopnost vytvářet větné vektorové reprezentace grafovou konvoluční sítí a relační grafovou konvoluční sítí.

3.4.6 Holmova metoda

Holmova metoda (22) je postup pro korekci p-hodnot při provádění více testů (testování více hypotéz). Je to vylepšená verze Bonferroniho korekce a je považována za méně konzervativní. Postup zahrnuje seřazení p-hodnot od nejmenší k největší a postupnou úpravu p-hodnot, přičemž každý test je porovnán s upravenou prahovou hodnotou. Tato metoda zajišťuje, že pravděpodobnost chyby prvního druhu (nepravé pozitivní) zůstává pod kontrolou při provádění více testů.

3.5 Grafová neuronová sítě

3.5.1 Využití

GNN (23) jsou silným nástrojem v oblasti strojového učení, zejména pro úkoly zahrnující data ve formě grafů. GNN je navržena tak, aby se učila reprezentace uzlů, hran a celých grafů, čímž zachycuje složité vztahy v síti. To činí GNN cennou pro širokou škálu aplikací, například pro klasifikaci uzlů, které mohou určit vlastnosti nebo přiřadit štítky jednotlivým uzlům v grafu. Například v analýze sociálních sítí mohou GNN být použity ke klasifikaci uživatelů na základě jejich interakcí. Také mohou být využity ke zpracování a pochopení textu, jelikož vazby v textech mohou být reprezentovány pomocí orientovaných grafů.

GNN jsou také velmi účinné pro úkoly založené na grafech. Například predikce spojení spočívá v předpovědi existence hran mezi uzly a při doporučování spojení v sociální síti. Kromě toho GNN vynikají v klasifikaci grafů, kde je cílem rozdělit celé grafy do předem definovaných tříd, například při klasifikaci molekul v oblasti objevování léků, či jako v tomto případě na klasifikaci vět.

Další důležitou aplikací GNN je doporučování v systémech založených na grafech, kde pomáhají zlepšit přesnost doporučení analýzou interakcí mezi uživateli a položkami. GNN se také používají pro obohacení znalostních grafů, kde pomáhají identifikovat chybějící vztahy mezi entitami. Celkově jsou GNN nezbytným nástrojem pro práci se složitými, vzájemně propojenými daty, které činí úkoly jako klasifikace uzlů, predikce spojení a klasifikace grafů efektivnějšími a účinnějšími.

3.5.2 Vysvětlení

GNN jsou typem neuronové sítě architektonicky přizpůsobeným k práci s grafy. Dokážou pracovat s nestrukturovanými neeuklidovskými daty. Podobně jako standardní neuronové sítě se skládají z vrstev:

- **Vstupní vrstva** přijímá počáteční reprezentace uzlů. V tomto výzkumu byly použity 300dimenzionální vektory. Matice vstupních dat lze reprezentovat jako $X \in R^{N \times F}$, kde N je počet uzlů a F je počet vstupních dimenzí.
- **Matice sousednosti** určuje vztahy mezi uzly. $A \in R^{N \times N}$, kde N je počet uzlů.
- **Skryté vrstvy** provádějí grafové operace, jako jsou konvoluce či agregace. Počet vrstev se obvykle pohybuje mezi 2 až 4 (v tomto výzkumu vždy 2). Transformace v každé vrstvě se provádí podle vzorce v rovnici 16.

Rovnice 16

$H^{(l+1)} = \sigma\left(AH^{(l)}W^{(l)}\right)$, kde: $H^{(l)}$ je stav uzlů v vrstvě l, A je normalizovaná matice sousednosti, $W^{(l)}$ jsou váhy, σ je aktivační funkce (funkce jedné reálné proměnné (Rectified Linear Unit, ReLU), sigmoid, atd.)}.

- **Výstupní vrstva** poskytuje konečnou reprezentaci grafu či uzlů. Počet dimenzí výstupu závisí na konkrétním úkolu. Při regresi je výstup skalární, zatímco při klasifikaci je počet dimenzí roven počtu tříd.

3.5.3 Předávání zpráv

Vrcholy (uzly) v GNN si předávají informace se sousedními uzly. Tento krok proběhne při každé konvoluci, což znamená, že jestliže mají sítě více skrytých vrstev, tak předají informace s větším množstvím uzlů. Každý uzel vytvoří zprávu na základě svého aktuálního stavu a odešle ji sousedním uzlům. Tímto procesem se shromáždí informace lokálního okolí a zachytí se struktura grafu.

Každý uzel $\boldsymbol{v}$ vytvoří zprávu $\boldsymbol{m_{u \to v}}$ pro své sousedy $\boldsymbol{u} \in \boldsymbol{N(v)}$ na základě svého aktuálního stavu $\boldsymbol{m_{u \to v} = f_m(h_u, h_v, e_{uv})}$, kde h_u a h_v jsou reprezentace (skryté stavy) uzlů, e_{uv} je hrana mezi uzly u a v, f_m je funkce generující zprávu (např. lineární transformace nebo MLP).

3.5.4 Agregace

Po obdržení zpráv od sousedních uzlů každý uzel agreguje tuto informaci a aktualizuje svou reprezentaci. Běžné agregační funkce zahrnují sumaci, průměrování nebo složitější neuronové metody. Výběr agregační funkce ovlivňuje schopnost modelu zachytit lokální vzory a závislosti v grafu. Pro dvě z našich GNN byla použita agregační funkce s váženým průměrem. Toto nám pomohlo zachytit i jednotlivé informace jako slovní druhy v struktuře grafu.

Každý uzel v po obdržení zpráv od sousedů aktualizuje svůj stav pomocí agregační funkce AGG. Viz rovnice 17.

Rovnice 17

$$h_v^{(k+1)} = \sigma\big(\mathrm{AGG}(\{m_{u \to v} \mid u \in \mathcal{N}(v)\})\big)$$

kde:

- $\boldsymbol{\sigma}$ je aktivační funkce (např. ReLU),
- $\boldsymbol{AGG}$ může být suma, průměr nebo vážený průměr, v našem případě se využili všechny tři možnosti:
- **Součet**: $\sum \boldsymbol{u} \in \boldsymbol{N(v)mu} \to \boldsymbol{v} \sum_{\boldsymbol{u} \in \boldsymbol{\mathcal{N}(v)}} \boldsymbol{m_{u \to v}} \sum u \in N(v) mu \to v$
- **Průměr**: $\frac{1}{|\mathcal{N}(v)|} \sum_{u \in \mathcal{N}(v)} m_{u \to v}$
- **Vážený průměr**: $\sum_{u \in \mathcal{N}(v)} w_{uv} m_{u \to v}$V

3.5.5 Transformace

Po agregaci je reprezentace uzlu transformována neuronovou vrstvou, často s použitím nelineární aktivační funkce jako ReLU. Tato transformace umožňuje uzlu učit se abstraktní rysy důležité pro daný úkol. V tomto výzkumu byla využita aktivační funkce ReLU pokaždé. Vzorec pro transfomaci v rovnici 18.

Rovnice 18

$$h_v^{(k+1)} = \sigma\left(W h_v^{(k+1)} + b\right)$$

kde W váhová matice, b je zkreslení, σ je aktivační funkce ReLU.

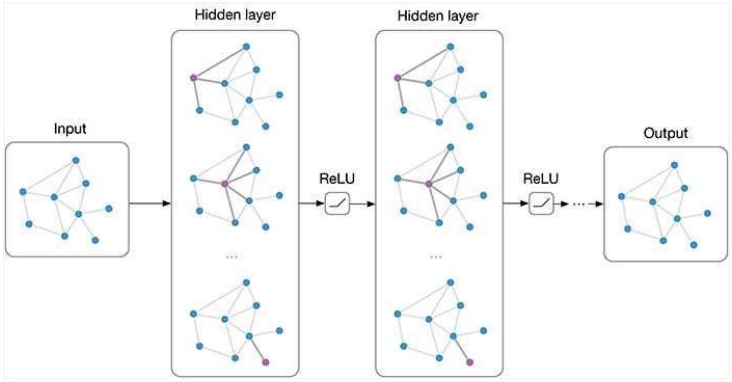
3.5.6 Výstupní funkce

Na konec proběhne výstupní funkce. Ta se ve všech GNN tohoto výzkumu/projektu lišila. V první, která se porovná, bylo využito základní zprůměrování. Poté bylo využito i složitější vážené, a na konec byla už čistá klasifikace. Poslední funkce však musí být permutací invariantní (např. součet nebo průměr vlastností uzlů), aby byla reprezentace grafu konzistentní bez ohledu na pořadí uzlů. Jde definovat jako v rovnici 19.

Rovnice 19

$$z_G = \text{READOUT}\left(\{h_v^{(K)} \mid v \in G\} \right)$$

kde READOUT představuje samotnou výstupní funkci. GNN se dá zobrazit jako na obrázku 3.



Obrázek 3: GNN se 2 skrytými vrstvami a aktivační funkcí ReLU (25)

3.5.7 Aktivační funkce ReLU

ReLU (23) je jedna z nejpoužívanějších aktivačních funkcí v neuronových sítích. Matematicky je vysvětlitelná jednoduše, jako $f(x) = max(0, x)$.

Toto znamená, že pokud je hodnota x kladná, výstup se nezmění, jestliže je záporná, tak je výstup 0. Jednoduchost funkce zrychluje procesy a řeší mizení gradientů. Také ale vzniknou

mrtvé neurony, které ztrácí schopnost se učit (stále záporné hodnoty). Výstup není na kladné straně omezen, což může způsobit nestabilitu jednotlivých modelů. Často se proto využívají upravené aktivační funkce jako je Leaky ReLU a Parametric ReLU.

3.6 Grafová konvoluční síť

Grafové konvoluční sítě (23) (GCN) rozšiřují koncept konvoluce z tradičních dat se strukturovanou mřížkou, jako jsou obrázky, na grafově strukturovaná data, což umožňuje modelům hlubokého učení pracovat v neeuklidovských doménách. Základní myšlenkou GCN je agregace informací z lokálního okolí uzlu, což umožňuje síti naučit se reprezentace, které zachycují jak vlastnosti uzlů, tak topologii grafu.

Standardní vrstva GCN je definována jako v rovnici 20.

Rovnice 20

$$H^{(l+1)} = \sigma\left(\widetilde{D^{-\frac{1}{2}}}\tilde{A}\widetilde{D^{-\frac{1}{2}}}H^{(l)}W^{(l)}\right)$$

kde $H^{(l)}$ je matice skrytých reprezentací uzlů v l-té vrstvě, $\tilde{A} = A + I$ je matice sousednosti s vlastními přidanými smyčkami, $\widetilde{D}$ je diagonální matice stupňů uzlů odpovídající $\tilde{A}$, $W^{(l)}$ je matice váhových parametrů, σ je nelineární aktivační funkce (např. ReLU).

Tato formulace vychází ze spektrální teorie grafů, kde se Laplacián grafu používá k lokálnímu vyhlazení vlastností uzlů. Normalizační trik $\widetilde{D^{-\frac{1}{2}}}\tilde{A}\widetilde{D^{-\frac{1}{2}}}$ zajišťuje numerickou stabilitu a zachovává měřítko.

GCN jsou obzvláště efektivní v úlohách polosupervizovaného učení, kde je označena pouze část uzlů v grafu. Iterativní agregací informací od sousedů mohou GCN predikovat štítky pro neoznačené uzly na základě struktury grafu i podobnosti vlastností.

Původní model GCN je omezen dosahem svého recepčního pole kvůli efektu nadměrného vyhlazení (oversmoothing) ve větších hloubkách. Mnoho rozšíření, jako jsou grafové attention sítě (GAT), grafové izomorfní sítě (GIN) nebo relační grafové konvoluční sítě (R-GCN), tyto limity překonává zavedením mechanismů pozornosti nebo expresivnějších agregačních funkcí.

3.7 Relační grafová konvoluční síť

Relační grafové konvoluční sítě (R-GCN) jsou rozšířením standardních grafových konvolučních sítí, které jsou navrženy pro práci s heterogenními grafy, kde uzly mohou být propojeny různými typy hran (relačními typy). Tradiční GCN modely se zaměřují na učení reprezentací pro homogenní grafy, kde všechny hrany mají stejný typ a váhu. R-GCN naopak modeluje různé vztahy mezi uzly. V této práci jsou tyto vztahy například větné členy.

Pro každý relační typ hrany má R-GCN vlastní váhovou matici, která zohledňuje specifické vztahy mezi uzly propojenými touto hranou. Relační grafová konvoluční síť (R-GCN) se liší od běžné GCN především způsobem, jakým předává zprávy mezi uzly. V R-GCN se zprávy mezi uzly váží podle typu vztahu mezi nimi. Tento přístup umožňuje modelu zachytit složité interakce mezi uzly založené na jejich specifických relačních vazbách.

Tato změna v předávání zpráv může být matematicky reprezentována v rovnici 21.

Rovnice 21

$$H_r^{(l+1)} = \sigma\left(\sum_{r' \in R} \widetilde{D^{-\frac{1}{2}} A_r D^{-\frac{1}{2}}} H^{(l)} W_r^{(l)}\right)$$

kde $H_r^{(l+1)}$ je matice skrytých reprezentací uzlů v l+1-té vrstvě pro relační typ r, $\widetilde{A_r}$ je sousednostní matice pro relační typ r (obsahuje informace o vztazích mezi uzly pro každý typ hrany), R je množina všech relačních typů, $W_r^{(l)}$ je matice váhových parametrů pro relační typ r, $\widetilde{D}$ je normalizační matice stupňů uzlů odpovídající $\widetilde{A_r}$, σ je nelineární aktivační funkce

Tato rovnice zajišťuje, že každá relační hrana mezi uzly je modelována odděleně, což umožňuje R-GCN lépe zachytit heterogenitu grafu.

R-GCN také umožňuje využívat vážené průměrování namísto běžného průměrování nebo maxování, což povoluje přesnější agregaci informací na základě váhy každé hrany. Tento přístup je obzvláště užitečný, pokud jsou v grafu různé typy vztahů, které mají různou důležitost pro konkrétní úlohu. Zároveň do váženého průměru či agregace, se mohou přidat typy uzlů.

3.8 Grafová porovnávací síť

Grafové porovnávací sítě (GMN) jsou navrženy pro výpočet skóre podobnosti mezi dvojicemi grafů tím, že společně uvažují o obou grafech pomocí mechanizmu křížové pozornosti. Na rozdíl od modelů grafových vnoření (graph embedding models), které každý graf mapují nezávisle na vektor, GMNs přímo vypočítávají podobnostní skóre mezi dvojicí grafů, což je činí účinnějšími pro úlohy určení podobnosti. V tomto výzkumu byly získány výsledky pomocí těchto porovnávacích sítí i pomocí modelů, které získaly vektor grafu na pozdější porovnání. Místa, kde se nejvíce liší od základních GNN jsou tyto:

3.8.1 Mechanismus křížové pozornosti:

GMNs využívají mechanismus křížové pozornosti, aby přiřadily uzly mezi grafy a identifikovaly rozdíly.

Tento mechanismus vypočítává pozornostní váhy, které měří, jak dobře uzel v jednom grafu odpovídá uzlům v druhém grafu.

Tyto váhy se používají k vytvoření vektorů křížového porovnání, které zachycují rozdíly mezi uzly v obou grafech.

3.8.2 Propagační vrstvy:

Podobně jako grafové neuronové sítě (GNNs), GMNs používají propagační vrstvy, které iterativně aktualizují reprezentace uzlů.

V každém kroku propagace modul aktualizace uzlů zohledňuje zprávy agregované z hran uvnitř grafu a také křížové vektory porovnání. Pro tyto propagační vrstvy se využil upravený model relační GCN, aby se získala, co nejpřesnější reprezentace mezi grafy, předtím než budou porovnány.

3.8.3 Výpočet skóre podobnosti:

Po několika krocích propagace jsou reprezentace uzlů agregovány do celkové reprezentace grafu.

Skóre podobnosti mezi dvěma grafy se následně vypočítá pomocí běžných metrik vektorového prostoru, v tomto výzkumu se využila kosinová podobnost.

4 METODOLOGIE EXPERIMENTŮ

4.1 Vážený word2vec

4.1.1 Předzpracování textu

V prvním modelu byl text zpracován v sedmi krocích. Prvním krokem byla tokenizace pomocí knihovny spaCy, která byla také využita na získání jednotlivých slovních druhů. Po přiřazení slovních druhů byly z textu odstraněny všechny symboly, na opravení gramatiky jednotlivých tokenů byl využit jednoduchý algoritmus. Z celého textu byly odstraněny stop slova. Poté byla jednotlivá slov vektorizována následovným způsobem, nejprve bylo nalezeno nejpodobnější slovo nacházející se v modelu, dále bylo otestováno slovo počáteční. Slova byla stemována a lematizována než bylo zjištěno, že žádné slovo v modelu není danému slovu podobné, slovu byl poté přiřazen vlastní vektor na základě slovního druhu.

4.1.2 Váhy

V této studii byl předpoklad vah pro slovní druhy v části s váženým word2vec, označených jako (w_{wt}), kde (wt) je index slovního druhu. Pro každý(w_{wt}) je předpoklad ($w_{\mathrm{wt}} \in Q$).

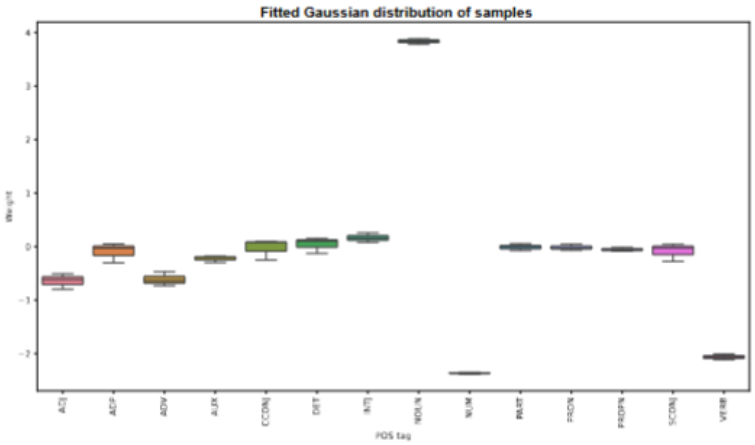
4.1.3 Vážení word2vec

Ze začátku byly vytvořeny nulové váhy pro každý slovní druh a následně byly použity objektové funkce. Opakovaně byla využita metoda BFGS. Velikost datasetu způsobila potřebu upravení začátečních podmínek před každým spuštěním, čímž byly získány rozdílné váhy. Metoda BFGS je totiž silně závislá na stavu těchto podmínek. Níže v tabulce 2 jsou příklady těchto vah.

Word type	Abbreviation	Weight - 1st iteration	Weight - 2nd iteration	Example word
Adjective	ADJ	1.000	1.000	last
Adposition	ADP	0.210	0.238	across
Adverb	ADV	0.903	1.066	separately
Auxiliary	AUX	0.415	0.396	would
Coordinating Conjunction	CCONJ	0.020	0.007	either
Determiner	DET	0.071	0.080	every
Interjection	INTJ	0.020	-0.006	oh
Noun	NOUN	-6.150	-6.651	brother
Numeral	NUM	3.470	4.467	five
Particle	PART	0.095	0.037	nt
Pronoun	PRON	0.085	0.100	somebody
Proper Noun	PROPN	-0.011	-0.585	Amrozi
Subordinating Conjunction	SCONJ	0.119	0.112	since
Verb	VERB	3.514	4.204	reported

Tabulka 2

Jelikož nezáleží na absolutních hodnotách vah jako spíše na jejich poměrech, nevadilo, že metodou BFGS byly získány váhy s rozdílnými hodnotami. Bez ztráty jakékoli informace byly pomocí Gaussovy distribuce zjištěny poměry mezi jednotlivými váhami. Přibližný vzhled distribuce lze vidět na obrázku 4.



Obrázek 4

Z Gaussovy distribuce poměrů vah vyobrazené na obrázku N bylo odebráno několik vzorků. Tyto vzorky byly mezi sebou testovány jako základní váhy pro Nelder-Mead metodu. Tato metoda často dosáhla lokálního maxima, tudíž nemohla být použita od začátku. Díky úpravě základních podmínek BFGS algoritmem měly váhy větší šanci dosáhnout globálního maxima.

Výše popsaným způsobem bylo získáno několik setů vah. Jednotlivé sety vah byly zkontrolovány validační částí datasetu. Váhy byly skalární čísla, tím pádem nenastal problém s overfitováním. Získané váhy mohly být použity v modelu definovaném i s jeho strukturou níže.

4.1.4 Výsledný model

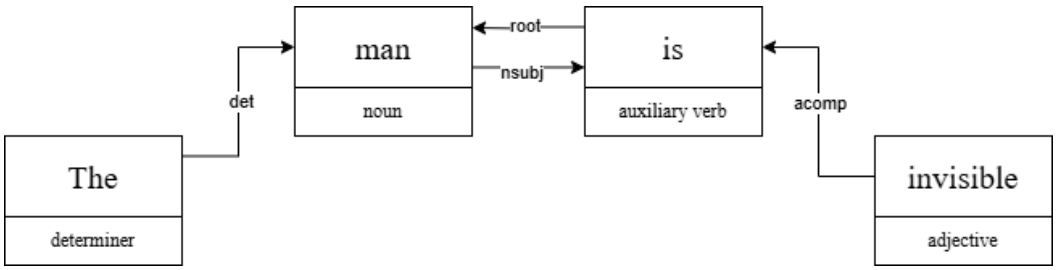
Z předzpracovaného text byly modelem prvně získány vektory a k nim přiřazené slovní druhy. Modelem byl proveden vážený průměr vztažený na všechny prvky. Tímto postupem byl získán průměrný vektor pro každou větu. Jednoduchostí tohoto postupu byla zajištěna větší rychlost modelu při zpracování textu oproti standartním modelům. Zároveň byla udržena poměrně kvalitní přesnost porovnatelná s nejlepšími BERT modely.

4.2 Grafové neuronové sítě

4.2.1 Předzpracování textu

V tomto modelu je text zpracováván mírně odlišně oproti modelu vážený word2vec, postup zpracování textu těchto dvou modelů je však shodný od kroku tokenizace po krok vektorizace. Po vektorizaci slov byly jednotlivé věty převedeny na grafy pomocí spaCy Postup na vytvoření grafů z vět probíhal pomocí knihovny spaCy. Prvně byly získány spojení mezi větnými členy (např. příslovce – podmět), které byly využity na sestavení grafů. V utvořených grafech byl

každým slovem reprezentován jeden uzel a spojení byly reprezentovány jako matice sousednosti. Ke spojením byl zároveň uložen jejich typ. Zároveň byla vytvořena další matice, reprezentující slovní druhy jednotlivých uzlů. Zhotovené grafy byly využity na vstup do našich sítí, s tím že rozdílné sítě využily rozdílný počet informací. Na grafu 1 je graficky znázorněna příkladová věta "The man is invisible."



Graf 1

Všechny grafy byly poté rozděleny do trénovací a validační části, pro kontrolu toho, že model nememoroval.

4.2.2 Váhy

V grafových neuronových sítí jsou váhy určeny několika způsoby. Matice vah je aplikována na vstupní rysy uzlů. Při vstupu má matice velikost u GCN, ve vstupní vrstvě 300 dimenzí, v první skryté vrstvě 384 dimenzí. Tato matice byla utvořena pro každou vrstvu a během trénování byla učena kombinovat informace od všech sousedů dohromady. Vrstvy u výstupu obsahují 192 dimenzí v poslední skryté vrstvě a 192 dimenzí ve výstupní vrstvě.

GCN využívá tzv. agregaci sousedních uzlů, tedy váhy aplikované na sousední uzly jsou stejné pro všechny uzly. Takto je zajištěno sdílení parametrů (tj. váhy pro všechny uzly jsou stejné, což pomáhá při trénování na různých grafech s různými strukturami).

R-GCN zavádí specifické váhy pro každý typ vztahu mezi uzly. Při num_relations různých typů vztahů jsou pro každý vztah přiřazeny vlastní váhy. V této práci bylo num_relations rovno 47. Ve vrstvě R-GCNConv se váhy liší podle typu vztahu mezi uzly, tedy váha pro vztah mezi dvěma uzly typu A by byla jiná než pro vztah mezi uzly typu B. Tato váha je typicky matice velikosti [počet vstupních dimenzí, počet dimenzí skryté vrstvy] pro každý vztah.

V této práci byla v modelu použita další váhová matice pro slovní druhy. Dimenze této matice byly určeny jako váhy pro vztahy mezi uzly. Slovních druhů bylo definováno 17 a větných členů 47.

4.3 Grafová konvoluční síť

4.3.1 Struktura

Grafová konvoluční síť (GCN) má relativně jednoduchou architekturu oproti pokročilejším modelům. Model GCN má celkem čtyři vrstvy s dimenzionalitou: (300, 384, 192, 128), tedy

obsahoval 213 504 parametrů. Touto strukturou je umožněno efektivní zpracování grafových dat a získávání reprezentací uzlů v grafu. Schéma architektury modelu je znázorněno na diagramu 1.

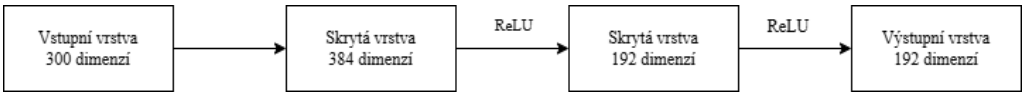


Diagram 1

4.3.2 Trénink

Trénink modelu byl proveden na datasetu QQP, který obsahuje páry otázek s cílem určit jejich významovou shodu. Po získání grafových průměrných vektorů byla aplikována kosinová podobnost. Vzhledem k tomu, že kosinová podobnost může nabývat hodnot v rozmezí -1 až 1, byla použita funkce sigmoid, kterou byly výstupy transformovány do intervalu (0,1). Výsledky byly porovnávány se skutečnými hodnotami pomocí ztrátové funkce binární křížové entropie s logity (BCEWLL).

Trénink byl proveden ve 24 epochách s optimalizátorem Adam, přičemž rychlost učení byla dynamicky upravována na základě validačních výsledků. Tímto přístupem byl stabilizován proces učení a byla zlepšena celková výkonnost modelu. Po dokončení tréninku byly výsledky ověřeny na testovacím datasetu, jejichž hodnocení je uvedeno v závěrečném odstavci.

4.4 Relační grafová konvoluční síť

4.4.1 Struktura

Model R-GCN se liší od základní GCN především v množství dodaných informací. V tomto modelu byly využity všechny informace, které byly do grafů uloženy. Druhy vztahů mezi uzly byly určeny na základě větných členů, přičemž jednotlivým slovním druhům byla přiřazena vlastní váhová matice. Tato váhová matice byla použita k agregaci informací a byla ní ovlivněna výstupní funkce.

Matematicky je tento vážený průměr definován v rovnici 22, kde H_j je zdrojový uzel a H_i je uzel cílový, W je matice vah, A_{dst} je váhová matice pro cílové uzle, N_{dst} je váhová matice definovaná typem uzlů, G je vektorová reprezentace grafu a $temperature$ upravuje sílu α.

Rovnice 22

$$\alpha = \mathrm{SoftMAX}\left(\frac{\mathrm{LeakyReLU}\left(\left(H_j \cdot W \cdot N_{\mathrm{dst}}\right) + \left(A_{\mathrm{dst}} \cdot H_i\right), 0.1\right)}{\mathrm{temperature}}\right)$$

$$G = \mathrm{Mean}\left(\alpha \cdot H_j\right)$$

Díky tomuto přístupu bylo modelu umožněno efektivněji pracovat s různými typy vztahů mezi slovy a větnými členy.

4.4.2 Trénink

Model byl trénován na datasetu QQP, ale na rozdíl od základní GCN měl značné problémy s overfitováním. Díky mnohonásobně většímu počtu vah byl model R-GCN schopen zapamatovat si všech 400 000 vět, protože byly správně nastaveny jeho hyperparametry.

Také byl zvýšen dropout a weight decay, čímž bylo zabráněno přeučení modelu. Hodnoty těchto parametrů byly nalezeny pomocí Bayesovské optimalizace, kterou bylo umožněno nalezení optimální kombinace hyperparametrů bez nutnosti manuálního ladění.

Další změna v tréninku se týkala ztrátové funkce. Místo BCEWLL byla použita funkce střední kvadratické chyby (MSE), kterou v tomto případě byly poskytnuty lepší výsledky. Tímto přístupem bylo modelu umožněno lépe rozlišovat jemné nuance mezi otázkami a bylo dosaženo vyšší přesnosti při klasifikaci jejich podobnosti.

Všechen trénink byl uskutečněn na osobním počítači, takže bylo nutné maximalizovat efektivitu operací. Bylo využito například dávkové učení a multiprocesování.

4.5 Grafová porovnávací síť

4.5.1 Struktura

Grafová porovnávací síť (GMN) měla nejsložitější strukturu ze všech implementovaných grafových neuronových sítí. Na počátku její architektury byla upravená verze našeho R-GCN modelu, která byla přizpůsobena tak, aby výstupní funkce dokázala efektivně vážit jednotlivé uzly v grafu. Hlavním rozdílem oproti předchozím modelům bylo, že GMN nejen zpracovávala informace na úrovni uzlů a hran, ale také prováděla porovnání mezi dvěma grafy.

Na počátku její architektury byla využita naše upravená verze **R-GCN**, která byla přizpůsobena tak, aby výstupní funkce efektivně vážila jednotlivé uzly v grafu. Tento model zpracovával informace nejen na úrovni uzlů a hran, ale také prováděl porovnávání mezi dvěma grafy.

Na rozdíl od předchozích modelů využívala GMN pouze **R-GCN** pro extrakci vlastností z grafu. Pro agregaci těchto vlastností a shrnutí informací byla použita **LSTM** vrstva, která umožňovala modelu zachytit složitější vztahy mezi grafy a lépe se vypořádat s různými strukturami dat.

Důležitou součástí modelu byla také vrstva pro porovnávání uzlů mezi dvěma grafy. Použila se metoda cosine attention, která vypočítala podobnost mezi uzly na základě jejich vektorových reprezentací. Výsledná pozornost byla následně kombinována s víceúrovňovou funkcí porovnávání (multi-perspective match function), která umožňovala zachytit různé aspekty podobnosti mezi uzly.

4.5.2 Trénink

Trénování GMN bylo výrazně složitější než u předchozích modelů kvůli její komplexní architektuře. Velký počet parametrů a složitější operace mezi grafy způsobovaly značné výpočetní nároky, což činilo trénink obtížným na dostupném hardwaru.

Abychom zabránili přeučení, byla aplikována kombinace několika technik. Dropout byl nastaven na vyšší hodnoty, weight decay byl zvýšen a byla použita **Bayesovská optimalizace** k nalezení optimálních hyperparametrů. Trénink probíhal na datasetu QQP a využíval různé způsoby agregace odpovídající složitosti modelu.

Další důležitou změnou oproti předchozím modelům bylo použití funkce **MSE** místo BCEWLL jako ztrátové funkce. Tato změna byla provedena na základě experimentů, které ukázaly, že MSE poskytuje lepší výsledky při porovnávání grafů.

Kvůli vysoké složitosti modelu bylo nutné rozdělit trénink na menší dávky, aby bylo možné efektivně využívat paměť GPU. Zároveň byla implementována dynamická úprava rychlosti učení podle validačních výsledků, což umožnilo stabilnější trénink a lepší generalizaci modelu na testovacích datech.

5 VÝSLEDKY

5.1 Vážený word2vec

5.1.1 Finální váhy

Výsledné váhy byly v některých případech záporné, přičemž podstatná jména měla nadměrně pozitivní váhu. Přídavná jména, podstatná jména, číslovky a slovesa měly největší váhy, zatímco jiné slovní druhy, například determinátory nebo předložky, měly váhy blízké nule. K tomu pravděpodobně došlo, protože tyto slovní druhy měly tak velký vliv na věty. Konečné váhy jsou zobrazeny v tabulce 3.

Word type abbreviation	Weight
ADJ	-1.330
ADP	0.341
ADV	-0.616
AUX	-0.334
CCONJ	0.126
DET	0.308
INTJ	-0.143
NOUN	4.970
NUM	-2.829
PART	-0.396
PRON	-0.060
PROPN	0.068
SCONJ	-0.011
VERB	-2.656

Tabulka 3

5.1.2 Klasifikace

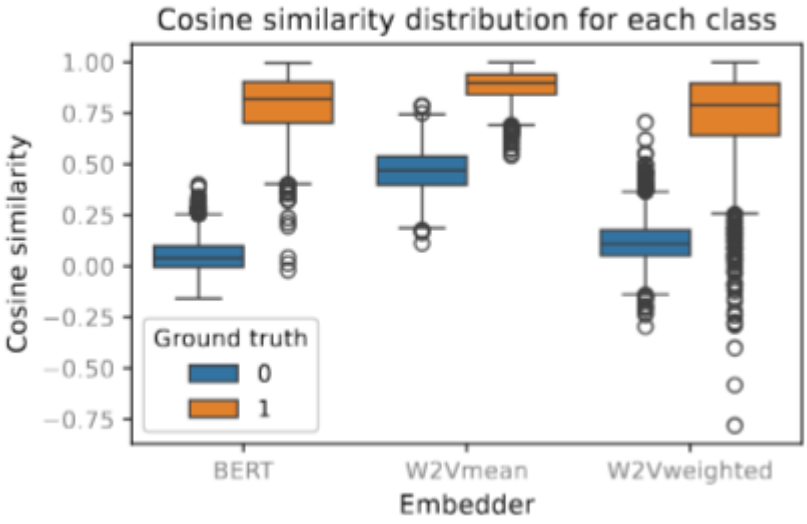
Byl porovnán přístup tohoto výzkumu s modelem BERT (Bidirectional Encoder Representations from Transformers) jemně doladěným pro vektorové reprezentace vět, konkrétně all-MiniLM-L12-v2, který má dobré výsledky v benchmarcích, a s jednoduchým průměrováním word2vec bez vážení. Všechny výsledky v tomto testu byly získány z nezávislého testovacího datasetu. Testovací dataset je vyvážený tak, že obsahuje stejné množství záznamů pro každou třídu (stejné a různé popisy). Pro měření statistik byla použita přesnost (Accuracy), F1 skóre a AUC. Viz tabulka 4.

Quality measure	Accuracy	F1 score	AUC
BERT	0.975767	0.975165	0.975767
W2Vmean	0.806442	0.837831	0.806442
W2Vweighted	0.933742	0.929273	0.933742

Tabulka 4

5.1.3 Statistiky

Výsledky jsou zobrazeny v tabulce a v boxplotu na obrázku 5. Vážené řešení podle typu slova přináší mnohem lepší výsledky než jednoduché průměrování. Vážené řešení má větší rozdíl mezi podobnými a odlišnými větami, ale není tak vysoký jako u modelu BERT. Vysoký výkon je pravděpodobně způsoben jeho architekturou, tréninkovými daty a kontextovým zpracováním celkového vstupu.



Obrázek 5

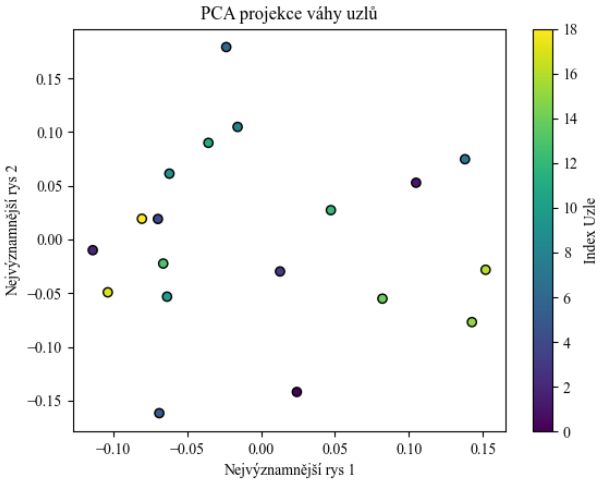
Rozdíly mezi zvažovanými embeddery byly testovány na významnost pomocí Friedmanova testu. Základní nulová hypotéza, že výsledky pro všechny 3 embeddry jsou stejné, byla silně odmítnuta, s dosaženou významností $p = 1.39 \times 10^{-297}$. Pro post-hoc analýzu byl použit Wilcoxonův test s podepsanými rangy s oboustrannou alternativou pro všechny páry porovnávaných embedderů, kvůli nekonzistenci běžnějšího post-hoc testu průměrných rangů s chybějícím uzavřeným světem v strojovém učení. Pro opravu vícehypotézového testování byla použita metoda Holm, která vedla k následujícím opraveným výsledkům:

- BERT vs. W2Vmean: $p = 4.01 \times 10^{-156}$
- BERT vs. W2Vweighted: $p = 2.12 \times 10^{-16}$
- W2Vmean vs. W2Vweighted: $p = 1.46 \times 10^{-183}$

5.2 Grafové neuronové sítě

5.2.1 Finální váhy

Sledování vah modelů grafových neuronových sítí je složitější, protože mají větší počet dimenzí. Byla vytvořena hlavní komponentní analýza finálních vah uzlů relační grafové konvoluční sítě, kterou si můžete prohlédnout na obrázku 6.



Obrázek 6

5.2.2 Klasifikace

Byly porovnány jednotlivé grafové neuronové sítě popsané v tomto výzkumu. Výsledky v tomto testu byly získány z nezávislého testovacího datasetu. Testovací dataset byl vytvořen rozdělením trénovacího, tak aby obsahoval stejné množství záznamů pro každou třídu. Pro měření statistik byla použita přesnost (Accuracy), F1 skóre a AUC.

	Accuracy	*F1 skóre*	*AUC*
GCN	0.78562	0.74021	0.85877
R-GCN	0.95125	0.93404	0.97707

5.2.3 Statistiky

Výsledky jsou zobrazeny v tabulce a v boxplotu na obrázku. Relační grafová konvoluční síť dosahuje mnohem lepších výsledků než grafová konvoloční síť. Rozdíl mezi významově stejnými a odlišnými větami je mnohem lépe definován relační R-GCN. Vyšší výkon je pravděpodobně způsoben využitím typů vztahů a uzlů.

Rozdíly mezi zvažovanými embeddery byly testovány na významnost pomocí Friedmanova testu. Základní nulová hypotéza, že výsledky pro 2 typy grafových konvolučních sítí jsou stejné, byla silně odmítnuta, s dosaženou významností $p = 4.71 \times 10^{-144}$.

6 ZÁVĚR

V úvodu práce bylo za cíl stanoveno zkoumání efektivity váženého word2vec pro embeddingy vět ve srovnání s průměrným agregováním a BERT. Bylo zjištěno, že složitá architektura neuronové sítě BERT poskytuje nejlepší výsledky. Také bylo zjištěno, že jednoduché průměrování bez vážení produkuje výsledky s nízkými rozdíly mezi cílovými třídami. Vážený word2vec produkuje podobně kvalitní výsledky jako model BERT, ve kterém snížení kvality bylo pouze 4 % při využití mnoho násobně méně zdrojů a rychlejšímu procesování dat.

Dále jsem si stanovil cíl otestovat zdali podobně kvalitní výsledky mohu replikovat pomocí grafových neuronových sítí a dokázat jejich kvalitu na získání vektorové reprezentace textu. Toto jsme dokázaly porovnáním s modelem BERT. Základní grafová konvoluční síť se nedokázala naučit dostatečný množství informací, aby byla o mnoho lepší než základní word2vec. V tu chvíli co se využily všechna větná data, která měla dostupná, Získala hodnoty, které mohly rivalovat samotnému modelu BERT. Samotný cíl se podle mě splnil tím, že jsme dokázaly tyto modely vytrénovat na osobním počítači a udržet kvalitu a přesnost.

Práce může pokračovat vytvořením modelu na vektorové zpracování celých odstavců. Model by využíval místo neuronovýc sítí na získání textových reprezentací grafové neuronové sítě.

7 BIBLIOGRAFIE

1. **Vasiliev, Yuli.** *Natural Language Processing with Python and spaCy: A Practical Introduction.* San Francisco : No Starch Press, 2020. 978-1-7185-0052-5.

2. **Srinivasa-Desikan, Bhargav.** *Natural Language Processing and Computational Linguistics: A Practical Guide to Text Analysis with Python, Gensim, spaCy, and Keras.* Birmingham : Packt, 2018. Sv. Expert Insight. 978-1-78883-853-5.

3. **Sebastian Raschka, Yuxi Liu, and Vahid Mirjalili.** *Machine Learning with PyTorch and scikit-learn: Develop Machine Learning and Deep Learning Models with Python.* Birmingham : Packt, 2022. 978-1-80181-931-2.

4. **Pecinovský, Rudolf.** *Python: Knihovny pro práci s daty pro verze 3.11.* Praha : Grada Publishing, 2023. Sv. Knihovna programátora. 978-80-271-0659-2.

5. **Ozsvald, Micha Gorelick and Ian.** *High Performance Python: Practical Performant Programming for Humans, Second Edition.* Beijing : O'Reilly, 2020. 978-1-492-05502-0.

6. **Steven Bird, Ewan Klein, and Edward Loper.** *Natural Language Processing with Python.* Beijing : O'Reilly, 2009. 978-0-596-51649-9.

7. *The Microsoft Research Paraphrase Corpus.* **Dolan, W. B., & Brockett, C.** 2005, Microsoft Research.

8. *On the over-memorization during natural, robust and catastrophic overfitting.* **Lin, Runqi, a další.** 2023, arXiv preprint arXiv:2310.08847.

9. **Turkington, Darrell A.** *Generalized vectorization, cross-products, and matrix calculus.* New York, USA : Cambridge University Press, 2013. ISBN 978-1-107-44872-8.

10. **Izrail‘ Moisejevič Gel‘fand, Mark E. Saul.** *Trigonometry.* Boston : Birkhäuser, 2001. 0-8176-3914-4.

11. *Efficient estimation of word representations in vector space.* **Mikolov, T., Chen, K., Corrado, G., & Dean, J.** 2013, OpenReview, stránky 1-12.

12. **Hastie, T., Tibshirani, R., & Friedman, J.** *The elements of statistical learning: Data mining, inference, and prediction.* 2. New York : Springer, 2009.

13. **Ian Goodfellow, Yoshua Bengio, and Aaron Courville.** *Deep Learning.* Cambridge, MA : MIT Press, 2016.

14. **Fletcher, R.** *FORTRAN subroutines for minimization by Quasi-Newton methods.* London : H.M. Stationery Office, 1972.

15. **Gabriel, D.** *Numerical solution of large displacement contact problems by the finite element method.* Praha : České vysoké učení technické, 2003. ISBN 80-01-02767-8.

16. **Leader, Jeffery J.** *Numerical analysis and scientific computation.* Boca Raton : CRC Press, 2022. ISBN 978-0-367-48686-0.

17. **Garnett, R.** *Bayesian optimization.* Cambridge, United Kingdom : Cambridge University Press, 2023. ISBN 978-1-108-42578-0.

18. **Kochenderfer, M. J., & Wheeler, T. A.** *Algorithms for optimization.* Cambridge, Massachusetts : The MIT Press, 2019. ISBN 978-0-262-03942-0.

19. **Bishop, Christopher M.** *Pattern recognition and machine learning.* New York : Springer, 2006. ISBN 978-0-387-31073-2.

20. **Myles Hollander, Douglas A. Wolfe, and Eric Chicken.** *Nonparametric Statistical Methods.* 3. Hoboken, New Jersey : Wiley, 2013. ISBN 978-0-470-38737-5.

21. **D'Abrera, E. L. Lehmann and H. J. M.** *Nonparametrics: Statistical Methods Based on Ranks. Rev. 1st ed.* New York : Springer, 2006. ISBN 0-387-35212-0..

22. **Peter H. Westfall, Randall D. Tobias, and Russell D. Wolfinger.** *Multiple Comparisons and Multiple Tests Using the SAS System.* Cary, North Carolina : SAS Institute, 2011. ISBN 978-1-59047-507-5.

23. **Zhou, Zhiyuan Liu and Jie.** *Introduction to Graph Neural Networks.* Cham : Springer, 2022. ISBN 978-3-031-00459-9.

8 SEZNAM OBRÁZKŮ A TABULEK