Západočeská univerzita v Plzni
Fakulta aplikovaných věd
Katedra informatiky a výpočetní techniky

Bakalářská práce

Full-textové vyhledávání s podporou porozumění textu dotazu

Plzeň 2019 Adam Mištera

Místo této strany bude zadání práce.

Prohlášení

Prohlašuji, že jsem bakalářskou práci vypracoval samostatně a výhradně s použitím citovaných pramenů.

V Plzni dne 2. května 2019

Adam Mištera

Poděkování

Děkuji především doc. Ing. Pavlu Královi, Ph.D. za odborné vedení, entuziasmus, cenné rady a ochotu, které mi v průběhu zpracování této bakalářské práce věnoval.

Abstract

This Bachelor Thesis aims to examine available methods of semantic representation along with the proposal of two own methods, which will be subsequently integrated into the already selected full-text search engine *Apache Solr*. At the same time, available data collections for full-text search will be examined in greater detail. The functionality of the methods will be subsequently verified on the selected data collection. The output of the work will be an evaluation of the obtained results, particularly the effectiveness of the methods leading to a greater accuracy and quality of searches in the selected data collection.

Abstrakt

Cílem této bakalářské práce je prozkoumat dostupné metody sémantické reprezentace současně s návrhem dvou vlastních metod, které budou následně implementovány do předem zvoleného full-textového vyhledávače *Apache Solr*. Současně budou podrobně prozkoumány dostupné datové kolekce pro full-textové vyhledávání. Funkčnost metod bude posléze ověřena na vybrané datové kolekci. Výstupem práce je zhodnocení dosažených výsledků, zejména účinnosti metod vedoucích ke zvýšení přesnosti a kvality vyhledávání ve zvolené datové kolekcí.

Obsah

1 Úvod

Rozšířením dostupnosti internetu na nejrůznějších zařízeních, zahrnujících osobní počítače, notebooky a především mobilní telefony, došlo k masivnímu navýšení počtu uživatelských dotazů každodenně zpracovávaných internetovými vyhledávači. Gigantem v této oblasti se stal v dnešní době nejpoužívanější vyhledávač *Google*, který měsíčně obslouží téměř 100 miliard dotazů, které prohledávají index obsahující přibližně 30 trilionů jednotlivých stránek [7, 16]. Uživatelsky očekávaným standardem se postupem času společně s rychlým obsloužením dotazu stává také forma dotazu, která se snaží co nejvíce přiblížit přirozenému jazyku.

Současně došlo k rozvoji umělých neuronových sítí - výpočetních modelů z oblasti umělé inteligence. Zvýšení výpočetního výkonu a úložné kapacity počítačů umožnilo implementaci náročných algoritmů, jejichž návrh započal již ve 40. letech minulého století [10]. Vzorem pro tyto algoritmy se stala biologická neuronová síť, například ta v lidském mozku, jejímž základním prvkem je neuron. V současné době získávají umělé neuronové sítě stále větší význam, neboť je možné naučit je řešit celou řadu velmi složitých problémů. Oblast uplatnění je velmi široká, sahá od samořídících aut, přes předpověď počasí a rozpoznání obrázků, až po zpracování přirozeného jazyka.

Cílem bakalářské práce je prostudovat dostupné metody sémantické reprezentace textu a na základě této studie navrhnout a implementovat full-textové vyhledávání s podporou porozumění textu dotazu, které se následně integruje do předem zvoleného full-textového vyhledávače *Apache Solr*. Výstupem práce bude zhodnocení získaných výsledků, respektive analýza účinnosti daných metod vzhledem ke zvýšení přesnosti a kvality vyhledávání ve vybrané datové kolekci.

V následující kapitole se seznámíme s metodami pro sémantickou reprezentaci textu. Zmíníme jejich dva nejvýznamnější zástupce, konkrétně systémy *Word2vec* a *FastText*, a u obou prostudujeme jejich klady a zápory. Posléze si představíme datové kolekce dostupné pro full-textové vyhledávání a analyzujeme full-textový vyhledávač *Apache Solr*. Kapitola 5 se zaměří na popis implementace a integrace vybraných metod sémantické reprezentace textu do vyhledávače *Apache Solr*. Následně uvedeme experimenty, jejichž cílem bude ověření funkčnosti navržených metod na zvolené datové kolekci. Závěr práce bude věnován shrnutí dosažených výsledků a navržení dalších vylepšení.

2 Sémantická reprezentace

Chceme-li se zabývat významem slov, popřípadě studiem obsahu složitějších jazykových útvarů jako jsou věty a souvětí, je potřeba si nejdříve pojem význam definovat. Podle výkladu cambridgeského slovníku je **význam** definován následovně: „The meaning of something is what it expresses or represents.“[1] Jedná se tedy o něco, co dané slovo vyjadřuje nebo reprezentuje.

Na první pohled jednoduchá definice ovšem skýtá řadu problémů. Za prvé, většina světových jazyků je poměrně bohatá na slovní zásobu a obsahuje i několik stovek tisíc různých slov. Současně se velká část těchto jazyků dále vyvíjí, čímž stále vznikají nové výrazy. Za druhé, nezanedbatelné procento slov jako například *láska* má abstraktní význam a nelze jej přesně definovat. Další, ne však poslední, problém tkví v existenci takzvaných slov souzvučných (*homonym*) a především slov mnohoznačných, například slovo *koruna* může představovat *královskou korunu*, *korunu stromu*, ale také *minci*. Právě vzhledem k těmto omezením, i přes mimořádně progresivní vývoj výpočetní techniky na přelomu tisíciletí, stále v drtivé většině případů není možné počítačům předložit slova ve formě nám známé z běžné konverzace, jejíž porozumění bylo umožněné jen díky mnoha tisícům let evoluce lidských mozků.

Počítač potřebuje slova převést do pro něj, pokud možno, pochopitelné, nejčastěji přijatelné numerické podoby. Nejjednodušším přístupem k tomuto kroku je vytvoření vektoru obsahujícího pouze binární hodnoty 0 a 1, takzvaného *one-hot vector* [6]. Jednička se ve vektoru vyskytuje pouze jednou, a to na místě vyhrazeném pro dané slovo z datového korpusu, přičemž zbytek vektoru vyplňují nuly.

$$\begin{aligned} \mathbf{u} &= \begin{bmatrix} 0\,0\,0\,0\,0\,0\,1\,0\,0\,0 \end{bmatrix}^{\mathrm{T}} \\ \mathbf{v} &= \begin{bmatrix} 0\,0\,1\,0\,0\,0\,0\,0\,0\,0 \end{bmatrix}^{\mathrm{T}} \end{aligned} \tag{2.1}$$

Kromě neefektivního využití místa, kde naprostou většinu vektoru zabírají nuly, má však tento přístup jeden zcela zásadní problém. Zvažme nyní následující jednoduchý příklad: n-rozměrný vektorový prostor obsahuje dva vektory $\mathbf{u}$ a $\mathbf{v}$, které můžeme vidět v rovnici 2.1. Vektor $\mathbf{u}$ zde představuje slovo *dům* a vektor $\mathbf{v}$ slovo *budova*. Víme, že se jedná o slova s velmi podobným významem, avšak vektory představující daná slova jsou ortogonální. V tomto konkrétním případě tedy nesdílí žádnou podobnost, a proto tento

[1] `https://dictionary.cambridge.org/dictionary/english/meaning`

jednoduchý přístup nelze využít pro tvorbu kvalitní sémantické reprezentace textu.

2.1 Distribuční hypotéza

Velká většina moderních metod pro zpracování přirozeného jazyka se úspěšně inspirovala výrokem slavného anglického lingvisty Johna R. Firthe

„You shall know a word by the company it keeps.“ [2]

Z tohoto výroku vyplývá, že slovo je určeno svým okolím. Distribuční hypotéza vycházející ze sémantické teorie přirozeného jazyka dále předpokládá, že slova, která se používají nebo vyskytují ve stejném kontextu, budou mít velmi pravděpodobně podobný význam [5]. Na základě okolí, ve kterém se dané slovo nachází, lze tedy do jisté míry určit jeho význam. Tento předpoklad se nyní pokusíme ukázat na následujícím příkladu.

velmi talentovaný fotbalista vstřelil v zápase branku
velice šikovný hokejista vstřelil během zápasu gól

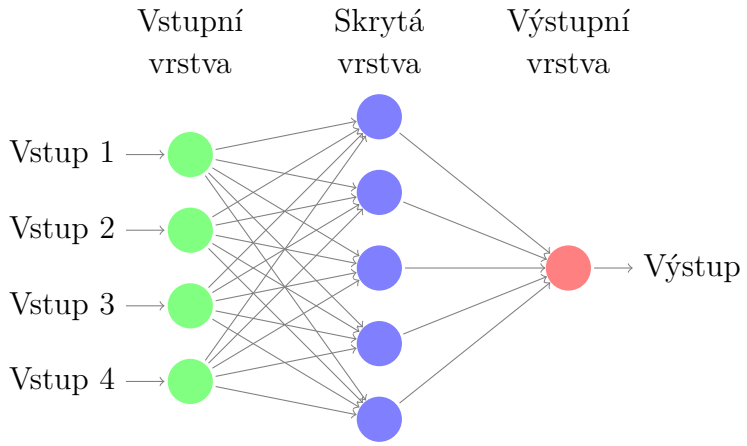
Na výše zmíněném příkladu můžeme vidět, že slova s podobným významem *fotbalista* nebo *hokejista* (a další zástupci jiných sportovních odvětví) mnohdy sdílejí obdobné okolí. V jejich okolí se vyskytují slova jako *zápas* nebo *gól*, dále také rozvíjející přídavná jména jako *talentovaný* nebo *šikovný*. V mluveném projevu i v literatuře se lze velmi často setkat s velkým množstvím slov souznačných nebo též synonym. Tato slova velkou měrou obohacují jazyk a rozšiřují slovní zásobu. Řadí se mezi ně, například *žena – dívka*, *krásný – nádherný* nebo také *jíst – konzumovat*.

Dalším příkladem, vhodně demonstrujícím platnost zmíněných tvrzení, může být věta „Šel do kavárny na . . . “, jejímž doplněním budou spíše slova *kávu* nebo *dort* než *představení* či *dovolenou*.

V následujících kapitolách 2.3 a 2.4 si projdeme dva modely, konkrétně *Word2vec* a *FastText*, založené na umělých neuronových sítích, které vychází z výše zmíněného předpokladu.

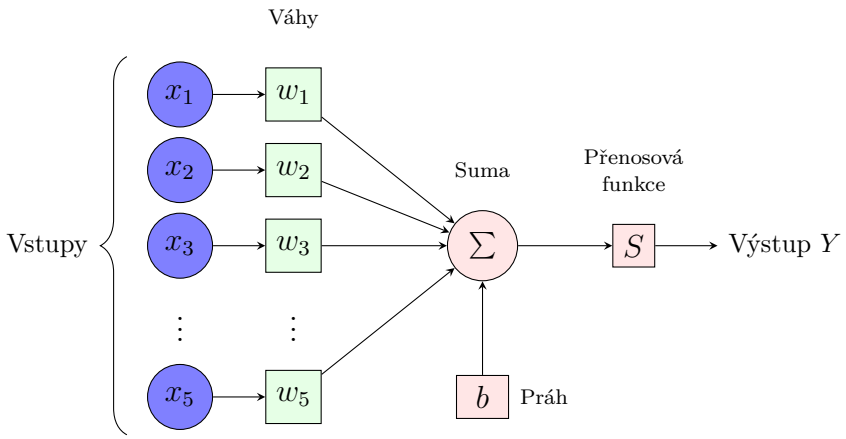
2.2 Umělá neuronová síť

Umělá neuronová síť (NN) je síť mnoha navzájem propojených neuronů, která modeluje skutečné neurony centrální nervové soustavy. Jejich návrh je inspirován vlastnostmi mozku a periferního nervstva [8].



Obrázek 2.1: Schéma umělé neuronové sítě

Na obrázku 2.1 můžeme vidět schéma umělé neuronové sítě. Síť se skládá ze tří základních vrstev. První je vstupní vrstva, která přijímá jednotlivé vstupní parametry. Tato neuronová síť dále obsahuje jednu skrytou vrstvu. Obecně může neuronová síť obsahovat více skrytých vrstev. Neuronové sítě obsahující více skrytých vrstev označujeme jako *hluboké* (anglicky *deep*), naopak ty s jednou skrytou vrstvou často jako *mělké* (anglicky *shallow*). Poslední vrstva je výstupní. Vrstvy jsou mezi sebou propojeny podobně jako neurony v lidském mozku.



Obrázek 2.2: Jednoduchý perceptron

Obrázek 2.2 zobrazuje umělý neuron, což je základní prvek tvořící neuronové sítě. Tento prvek se často nazývá také *perceptron* [8]. Ve své podstatě se jedná o nejjednodušší model umělé neuronové sítě sestávající pouze z jednoho neuronu. Perceptron je binární klasifikátor, který mapuje vektor vstupů

$\mathbf{x} = [x_1, x_2, \ldots, x_n]$ na výstup neuronu Y podle rovnice 2.2

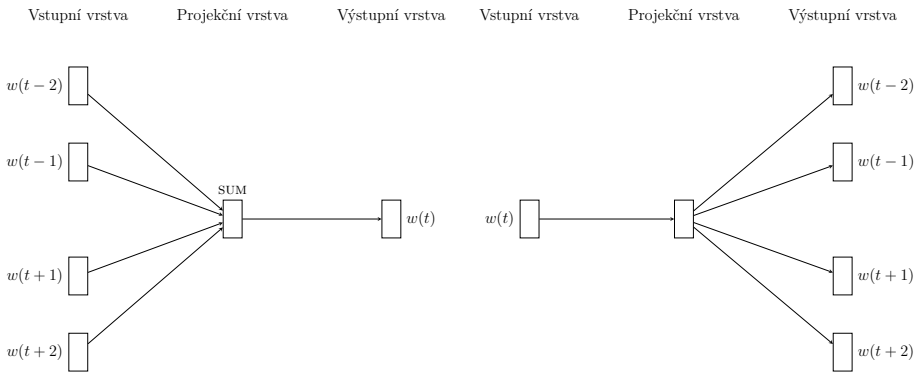
$$Y = S(\sum_{i=1}^{N} w_i \cdot x_i - b), \tag{2.2}$$

kde $\mathbf{w} = [w_1, w_2, \ldots, w_n]$ je vektor vah a b je konstanta nazývaná jako práh (anglicky *bias*). Symbol $S(\mathbf{x})$ označuje přenosovou funkci neuronu (jinak také aktivační funkce). Je-li skalární součin $\mathbf{w} \cdot \mathbf{x}$ menší než práh, zůstává neuron v pasivním stavu.

V praxi se využívá několik různých druhů přechodových funkcí, často používanou je například sigmoidální přenosová funkce.

2.3 Word2vec

Word2vec [11, 12] je skupina modelů založených na umělých neuronových sítích určených pro vytvoření slovních vektorů (anglicky *word embedding*). Vstupem modelu *Word2vec* je obsáhlý soubor textů vybraného jazyka, jinak také zvaný jazykový korpus. Příkladem vhodného datového korpusu je internetová encyklopedie *Wikipedia*, která v anglické mutaci obsahuje několik milionů různých článků[2] dostačujících pro získání kvalitního výstupu. Proces vytvoření vnoření slov spočívá v převedení frází, popřípadě slov jazykového korpusu na vektory umístěné ve vektorovém prostoru většinou čítajícím stovky dimenzí. Jednotlivým slovům poté odpovídá vždy jeden vektor z tohoto prostoru. Slovní vektory sdílející podobný či blízký význam jsou umístěny blízko sebe. Naproti tomu nesouvisející slova či slova opačného významu jsou ve vektorovém prostoru daleko od sebe.



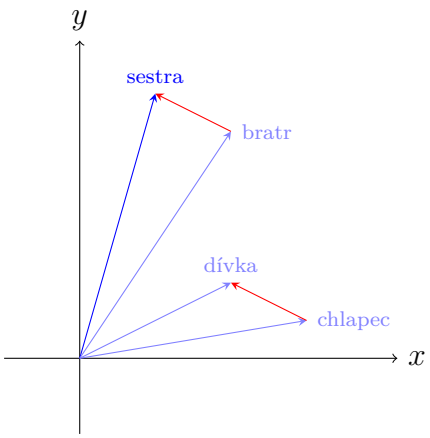
(a) CBOW (b) Skip-gram

Obrázek 2.3: Druhy modelu *Word2vec*

[2]Česká verze encyklopedie obsahovala v roce 2018 přes 400 000 článků.

Model *Word2vec* můžeme dále rozdělit na dvě podskupiny, které jsou pojmenované *Skip-gram* a *Continuous Bag-of-Words* (*CBOW*). *CBOW* se snaží na základě okolí předpovědět dané slovo. Oproti tomu *Skip-gram* využívá konkrétního slova, aby pomocí něj předpověděl slova vyskytující se v jeho okolí. Obě podskupiny můžeme vidět znázorněné na schématu 2.3.

Jednoduchým znázorněním demonstrujícím praktický význam této vlastnosti může být příklad, kdy odečtením slovního vektoru *bratr* od vektoru *chlapec* a přičtením vektoru *dívka* získáme vektor odpovídající slovu *sestra*. Vizualizaci tohoto faktu můžeme vidět na obrázku 2.4. Červeně znázorněné vektory odpovídají rozdílu mezi slovními vektory *chlapec* a *dívka*.



Obrázek 2.4: Vektor `bratr` + `chlapec` − `dívka` = `sestra`

Pomocí modelu *Word2vec* můžeme zkoumat i jiné vztahy, jako například vztah hlavní město – stát odpovídající slovnímu spojení Praha – Česká republika. Využitím modelu korektně natrénovaného na dostatečně velkých datech jsme schopni tyto vztahy (nejen) odhalit, ale také následně využít dále.

Dokument nebo soubor dokumentů předzpracovaný tímto způsobem je posléze možné využít jako vstup pro další algoritmy z oblasti strojového učení zabývajícího se zpracováním přirozeného jazyka. Mezi ně se řadí také algoritmy pro analýzu sentimentu [17], konkrétně například rozhodnutí, zda je příspěvek na sociálních sítích pozitivní či negativní. Dále sem patří kategorizace textu, tj. zařazení článku do odpovídajících kategorií jako jsou například sport, kultura, politika a podobně.

2.4 FastText

FastText [6, 14] je rozšíření modelu *Word2vec* navržené v roce 2016 firmou *Facebook* provozující největší[3] sociální síť na světě. Na rozdíl od modelu *Word2vec*, který považuje každé slovo v korpusu za atomickou jednotku, *FastText* považuje slovo za množinu několika n-gramů. Pojem n-gram je definován jako řetězec prvků (jako například slova či písmena), které se objevují v delší posloupnosti[4]. Například trigramy slova *jablko* jsou *jab*, *abl*, *blk* a *lko*. Výsledný slovní vektor slova *jablko* je poté součtem vektorů jednotlivých n-gramů.

Díky této vlastnosti získává *FastText* oproti zmíněnému *Word2vec* několik výhod. První z nich je kvalitnější systém reprezentace slov, která se ve vstupních textech vyskytují jen zřídka. Přestože se v textu tato slova objevují jen vzácně, jejich n-gramy jsou sdílené i s ostatními slovy obsaženými ve zpracovávaných dokumentech. Druhou důležitou vlastností je schopnost zkonstruovat vektor i pro slova, která se ve vstupních textech vůbec nevyskytují z n-gramů jiných slov, čehož model *Word2vec* není schopen.

Nevýhodou tohoto řešení je zvýšená paměťová i časová náročnost vzhledem k potřebě zpracovávat jednotlivé n-gramy. Z tohoto důvodu je nutné být vybaven dostatečně výkonným hardwarem, což však v dnešní době není neřešitelný problém.

2.5 Alternativní modely

Mezi další modely, které je možné využít pro sémantickou reprezentaci textu, se řadí *Glove* [13]. Jedná se o open-source projekt vyvíjený od roku 2014 na Stanfordově univerzitě. Proces vytvoření slovních vektorů spočívá v sestavení matice vzájemného výskytu (anglicky *co-occurence matrix*), kterou označíme X. Prvky matice X_{ij} udávají frekvenci výskytu slova j v kontextu slova i. Vzhledem k vysokému počtu jednotlivých kontextů je posléze provedena faktorizace matice X a řádky odpovídají samotným slovním vektorům.

2.6 Metriky

Pro vyhodnocení podobnosti vektorů, respektive určení jejich vzdálenosti lze využít několik různých metrik. V následující části si vybrané metriky

[3] *Facebook* podle údajů z ledna 2019 registruje přes 2 271 milionů uživatelských účtů, tedy téměř o 400 milionů více než druhá největší sociální síť *YouTube* [15]

[4] `https://en.oxforddictionaries.com/definition/n-gram`

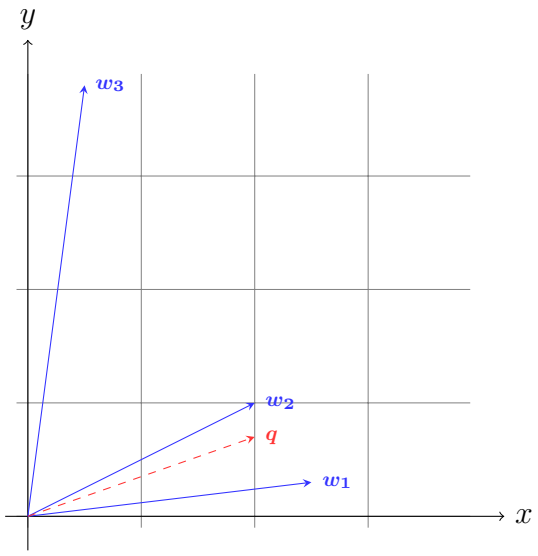
podrobně představíme a zvolíme nejvýhodnější z nich.

2.6.1 Eukleidovská metrika

V n-rozměrném eukleidovském prostoru můžeme spočítat vzdálenost d mezi dvěma body $\mathbf{p} = (p_1, p_2, \ldots, p_n)$ a $\mathbf{q} = (q_1, q_2, \ldots, q_n)$ definovanou podle rovnice 2.3.

$$\begin{aligned} d(\mathbf{p}, \mathbf{q}) &= \sqrt{(p_1 - q_1)^2 + (p_2 - q_2)^2 + \cdots + (p_n - q_n)^2} \\ &= \sqrt{\sum_{i=1}^{n} (q_i - p_i)^2} \end{aligned} \tag{2.3}$$

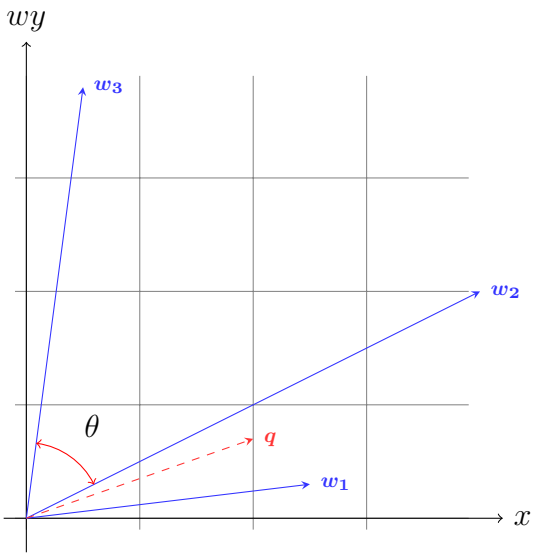
Výše uvedená metrika je pojmenovaná podle slavného řeckého matematika a geometra žijícího na přelomu 4. a 3. století před naším letopočtem. Na obrázku 2.5 můžeme vidět ukázkový zjednodušený dvourozměrný vektorový prostor, do kterého jsou zobrazena slova, reprezentovaná vektory w_1, w_2, w_3 a dotaz (porovnávané slovo) q. Z hlediska eukleidovské vzdálenosti leží nejblíže k dotazu q vektor w_2.



Obrázek 2.5: Dvourozměrný vektorový prostor

Nyní však zvažme následující situaci. Vektor w_2 prodloužíme a zdvojnásobíme jeho délku. Směr vektoru se v tomto případě nijak nezměnil, avšak nyní je ke slovu q nejblíže vektor w_1, jak je zachyceno v grafu na obrázku 2.6.

Z tohoto důvodu se v praxi v některých případech můžeme setkat s použitím jiných metrik.



Obrázek 2.6: Dvourozměrný vektorový prostor (prodloužený vektor)

2.6.2 Kosinová podobnost

Jedním z příkladů je **kosinová míra**, která bere v úvahu úhel, jenž mezi sebou dva vektory svírají. Intuitivně vzato, čím větší úhel dva vektory svírají, tím jsou od sebe vzdálenější. V našem případě by tato skutečnost poukazovala na fakt, že se význam vybraných slov liší. Například slovo *auto* bude mít blíže ke slovům *automobil* nebo *vozidlo* než například ke slovu *pes*. Tato norma však nebere v potaz případnou normalizaci daných vektorů.

Máme-li dané dva vektory **A** a **B**, poté můžeme **kosinovou míru** definovat pomocí velikosti vektorů a skalárního součinu dle rovnice 2.4 jako

$$podobnost = \cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}}, \tag{2.4}$$

kde A_i a B_i reprezentují složky vektoru A, respektive B. Úhel θ je naznačen na obrázku 2.6 mezi vektory w_3 a w_2.

2.6.3 Manhattanská metrika

Další možnou metrikou pro výpočet vzdálenosti dvou vektorů je například také **Manhattanská metrika** (často též označovaná jako newyorská metrika) pojmenovaná podle pravoúhlého systému ulic na Manhattanu, jednoho z pěti světově proslulých newyorských městských obvodů. Manhattanská me-

trika je na množině $\mathbb{R}^n$ definovaná vztahem

$$\begin{aligned} d(\mathbf{p},\mathbf{q}) &= |p_1 - q_1| + |p_2 - q_2| + \cdots + |p_n - q_n| = \|\mathbf{p} - \mathbf{q}\| \\ &= \sum_{i=1}^{n} |p_i - q_i| \end{aligned} \tag{2.5}$$

Její uplatnění lze nalézt například během šachové partie, kde je pro určité figury (zejména *věž*) vzdálenost, kterou musí urazit mezi políčky, spočtena právě pomocí této metriky. Její využití je tedy především právě pro pravoúhlé systémy. Z tohoto důvodu je však tato metrika pro naše účely nevhodná, a proto budeme dále využívat jen kosinovou míru.

3 Datové kolekce

Klíčovým krokem pro ověření funkčnosti metod navrženého systému pro vyhledávání informací (anglicky *information retrieval* zkráceně IR) je nalezení vhodné **datové kolekce**. Vytvoření takové kolekce je časově náročný úkol, jelikož po sběru potřebných dat je nezbytné tyto kolekce anotovat, respektive je nutné ručně popsat a tím přiřadit relevantní data k jednotlivým tématům a oddělit od těch, která se témat netýkají. Obsahem této kapitoly bude představení tří datových kolekcí vhodných pro vyhledávání informací a následný výběr té nejvhodnější pro vypracování bakalářské práce.

3.1 Reuters

První zmíněnou kolekcí je **Reuters-21578**[1]. Uvedená datová kolekce je běžně používaná v mnoha oblastech počítačového zpracování přirozeného jazyka (anglicky *natural language processing* zkráceně NLP). Kolekce obsahuje 21 578 novinových článků napsaných v anglickém jazyce a vydaných během roku 1987 celosvětově známou zpravodajskou agenturou *Reuters*.

Kolekce sestává z 22 datových souborů ve formátu značkovacího jazyka SGML. Až na drobné odchylky je tato struktura totožná s XML. Každý ze souborů obsahuje přesně 1000 článků vyjma posledního, kde je obsaženo zbývajících 578. Jednotlivé články jsou umístěny ve značce (anglicky *tag*) nazvané REUTERS, kde je dále specifikováno datum vydání článku, jeho název a především text. Vybrané články obsahují také upřesnění tématu, jehož se týkají, dále osob, které jsou součástí hlavního sdělení článku, popřípadě místa, kde událost nastala.

Přes širší možnosti využití této kolekce je však její primární určení ke kategorizaci textu.

3.2 TREC

Text REtrieval Conference (**TREC**) [2] umožňuje pro vyhledávání informací využít velké množství datových kolekcí ze specifických oblastí, jako je například chemie, lékařství (konkrétně systém pro podporu rozhodování, anglicky

[1] `http://kdd.ics.uci.edu/databases/reuters21578/reuters21578.html`

[2] `https://trec.nist.gov/data.html`

clinical decision support system zkráceně CDDS) nebo také genetiky (datové kolekce obsahují genomy) a mnoha dalších vědních disciplín.

Jednotlivé datové kolekce jsou nejčastěji strukturované ve formátu XML a převážně psané v anglickém jazyce. Lze se však setkat také s arabštinou, čínštinou a španělštinou. Všechny datové kolekce obsahují seznam relevantních témat k jednotlivým článkům, který lze následně využít pro vyhodnocení získaných výsledků za pomoci programu `trec_eval`.

3.3 CLEF

Poslední zde zmíněná kolekce byla vytvořena uskupením Conference and Labs of the Evaluation Forum (**CLEF**). Konkrétně je datová kolekce pojmenována *CLEF AdHoc-News Test Suites (2004-2008)*[3]. Jedná se o rozsáhlý soubor vícejazyčných dokumentů v textové podobě, navržených přímo pro potřeby vyhodnocení kvality vyhledávání informací. Kompletní balíček obsahuje kolekce novinových článků z různých světových periodik v mnoha jazycích (čeština, angličtina, němčina, španělština a další) vydávaných mezi lety 2004 a 2008. Mezi české zdroje se řadí články z *Mladé Fronty* (68 842 článků) nebo z *Lidových novin* (12 893 článků). Ke článkům je zároveň přiřazeno 50 různých témat z oblastí jako kultura, sport, události ve světě či politika. Daná témata se posléze využijí k sestavení vyhledávaných dotazů. Seznam všech témat je dostupný v příloze B.

3.3.1 Struktura korpusu

Data jsou v rámci jednotlivých textových souborů strukturována především ve formátu značkového jazyka XML. Jednotlivé jazyky mají více či méně stejných tagů, mezi základní se řadí - DOCNO, DOCID, DATE, TITLE a TEXT. Novinové články jsou v tagu DOC; další tagy obsahují datum vydání článků, jeho identifikátor, název a text.

Součástí balíčku jsou také témata a takzvaná ohodnocení (anglicky *assessment*), kde je označena relevantnost článků pro daná témata. Dále je přiložen program `trec_eval` sloužící pro automatické ověření kvality navrženého systému.

3.3.2 Program `trec_eval`

Součástí datové kolekce je také výše zmíněný program `trec_eval`, který slouží k automatickému vyhodnocení výsledků získaných z informačního sys-

[3] `http://catalog.elda.org/en-us/repository/browse/ELRA-E0036/`

tému. Vstupem aplikace jsou dva soubory. Obsahem prvního je seznam relevantních dokumentů pro jednotlivá témata, popřípadě dotazy, druhý soubor obsahuje výsledky získané z informačního systému.

Program lze spustit i s dalšími parametry. Zadání parametru `-m` umožňuje konkrétně specifikovat sledované míry, eventuálně lze také nastavit vyhodnocení pro každé téma odděleně pomocí parametru `-q`.

Výstup programu tvoří vyhodnocení vyhledávání, jehož důležitou součástí je počet navrácených článků, relevantních článků, navrácených relevantních článků a velké množství metrik. Při spuštění programu s parametrem `-a` lze vyhodnotit více než 130 různých metrik.

3.4 Zvolená datová kolekce

Z uvedených datových kolekcí jsme pro vypracování bakalářské práce zvolili **CLEF**. Hlavním důvodem bylo uzpůsobení této kolekce přímo pro účely *vyhledávání informací*. Rozhodujícím parametrem byla také velikost kolekce a její vícejazyčnost (především přítomnost českého jazyka). Dalším důvodem byla dostatečná obecnost této kolekce oproti databázi **TREC**, která je velmi úzce zaměřená.

4 Full-textové vyhledávání

Vzhledem k neustálému navyšování objemu dat internetu je potřeba tato data pro zajištění rychlého přístupu a efektivnější manipulaci skladovat a indexovat. Mezi nejvýznamnější zástupce aplikací zaměřujících se na tuto problematiku se řadí především vyhledávač *Apache Solr* [4] a jeho konkurent *Elasticsearch* [3]. Oba projekty jsou založené na open-source knihovně *Apache Lucene* [9]. Současně jsou také oba vyhledávače napsané v programovacím jazyce *Java* a poskytují velice podobné funkce. *Apache Solr* je však starší, jeho vývoj započal již v roce 2004. Oproti tomu vyhledávač *Elasticsearch* byl vytvořen až v roce 2010. Pro účely této práce byl z důvodu požadavků zákazníka zvolen systém *Apache Solr*.

4.1 Apache Solr

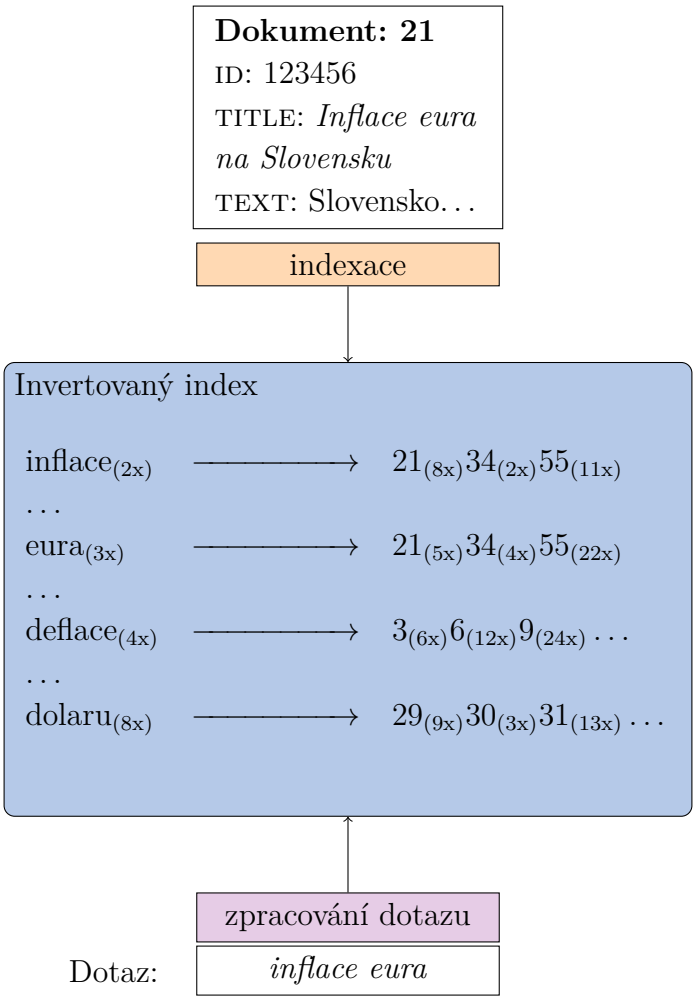
Apache Solr [4, 9] je open-source platforma napsaná v jazyce *Java*, která byla navržená a optimalizovaná pro full-textové prohledávání a indexování rozměrných textových dat.

Na rozdíl od většiny běžných databázových systémů, kde je k jednotlivým dokumentům přiřazeno pole slov, která jsou součástí dokumentu, *Apache Solr* používá odlišnou metodu, takzvaný **invertovaný index** [4]. Invertovaný index ke každému slovu nebo výrazu obsaženému v korpusu mapuje všechny dokumenty, ve kterých je obsažen.

4.1.1 Invertovaný index

Na obrázku 4.1 můžeme vidět schéma, zobrazující, jak funguje invertovaný index používaný full-textovým vyhledávačem *Apache Solr*. Nejdříve, jak je naznačeno v horní části schématu, je zvolený dokument indexován pomocí vyhledávače *Solr*. Indexovanému dokumentu je nejprve vnitřním systémem *Solr* přiřazeno identifikační číslo. Posléze jsou během fáze indexování nalezeny všechny jedinečné výrazy vyskytující se v dokumentu společně s četností jejich výskytu.

Světle modrý obdélník obsahuje všechny zaindexované položky, jedná se tedy o samotný invertovaný index. Na levé straně jsou uvedena jednotlivá slova společně s celkovým počtem dokumentů, které je obsahují. Na pravé straně jsou uvedeny dokumenty, ve kterých se slovo vyskytuje zároveň s počtem výskytů v těchto dokumentech.



Obrázek 4.1: Invertovaný index

Ve spodní části obrázku je naznačeno vyhledávání pomocí *Solr*. Uživatel v tomto případě hledá ukázkový výraz *inflace eura*. Vyhledávač uživateli vrací seznam všech dokumentů, kde se tyto výrazy vyskytují, konkrétně se jedná o dokumenty s indexem 21, 34 a 55.

4.1.2 Indexování dat

Vyhledávač *Apache Solr* umožňuje data pro indexování vkládat v mnoha běžně používaných formátech jako XML nebo CSV. Soubory však musí dodržovat předepsanou normu, takže je v určitých případech nutná úprava. Výhodou je rozmanitost dat, která dokáže *Solr* indexovat. Lze nastavit, jaká data budou pole obsahovat, ať se jedná o data v textové podobě, numerické hodnoty v pohyblivé řádové čárce, popřípadě částky v různých měnách.

Apache Solr indexuje data v oddělených jádrech (anglicky *core*). Z tohoto

důvodu je před samotnou indexací nejdříve potřeba vytvořit jádro pomocí příkazu `bin/solr create -c <jméno>`. Posléze lze dokumenty zaindexovat příkazem `bin/post -c <jméno> <umístění_souboru>`. Poté jsou všechny dokumenty umístěné v souborech zaindexovány a připraveny k prohledávání.

4.1.3 Práce s vyhledávačem *Apache Solr*

Pro práci s vyhledávačem *Apache Solr* můžeme využít webovou administraci dostupnou ve většině případů po spuštění ukázkového projektu otevřením prohlížeče na adrese `http://localhost:8983/solr/#`. Pomocí vyhledávače *Apache Solr* můžeme nad libovolně zvolenými a zaindexovanými daty provádět dotazy (anglicky *query*). S použitím dotazu lze rychle získat požadovaná data s výstupem formátovaným dle potřeby. Při vyhledávání lze aplikovat nejrůznější filtry pro zpřesnění výsledků nebo omezení rozsahu oblasti prohledávání dotazu.

Další a zároveň velmi efektivní možností, jak provádět dotazy nad vyhledávačem *Apache Solr*, je sestavení požadovaného dotazu společně se zadáním potřebných parametrů a následně jeho spuštěním z příkazové řádky pomocí nástroje `curl`[1].

4.1.4 Vyhledávání pomocí *Apache Solr*

Ukázkový dotaz může být poté formulován například jako:

```
http://localhost:8983/solr/core/select?q=text:pes AND text:sluch
```

Tento dotaz vyhledá všechny dokumenty, které ve svém textu obsahují výrazy *pes* a zároveň *sluch*. Uživatel by se patrně tímto dotazem pokoušel vyhledat články nebo informace týkající se kvality psího sluchu.

Relevance nalezených dokumentů je určena pomocí interního ohodnocení, takzvaného bodování (anglicky *scoring*). Výsledek vyhledávání je seřazen od dokumentů s nejvyšším hodnocením až po ty s nejnižším. Hlavní parametry udávající bodování dokumentu jsou *Term Frequency* (TF) a *Inverse document frequency* (IDF). TF udává počet opakování slova v daném dokumentu a IDF převrácenou četnost slova ve všech dokumentech.

Hodnocení lze dále ovlivnit přidáním symbolu stříšky a požadované numerické hodnoty, například `q=text:pes^2`. Tímto lze zvýšit důležitost zvolených výrazů (alternativně lze váhu také snížit), které poté zvýší výsledné hodnocení dokumentu, v němž se vyskytují.

[1] Obecně lze pro vyhledávání použít libovolný **REST** klient.

4.1.5 Výstup vyhledávání

Výstup vyhledávání lze téměř libovolně upravit dle potřeb uživatele. V základním nastavení vyhledávač *Apache Solr* vrací nalezené dokumenty ve formátu JSON. Pro naše potřeby jsme výstup upravili do podoby dokumentu CSV s hodnotami oddělenými pomocí mezer. Výsledný formát byl totožný s formátem požadovaným programem `trec_eval` pro vyhodnocení kvality vyhledávání viz kapitola 3.3.2.

5 Návrh a implementace řešení

Cílem této kapitoly bude představit řešení integrace metod sémantické reprezentace textu do vyhledávače. Konkrétně se zaměříme na sestavení rozšířeného dotazu, který bude použit v prostředí *Apache Solr*.

5.1 Evaluační metriky

Při vyhodnocení výsledků kvality námi navrženého systému s již integrovanými metodami pro sémantickou reprezentaci textu jsme sledovali tři hlavní míry, které jsou v praxi běžně používané především v oblasti získávání informací. Jedná se o *přesnost* (anglicky *precision*), *úplnost* (anglicky *recall*) a zejména *F-míru* (anglicky *F-measure*). *Přesnost* nám určuje, kolik vybraných dokumentů je relevantních z celkového počtu navrácených dokumentů. Druhá zmiňovaná míra *úplnost* nám pomáhá určit procento navrácených relevantních dokumentů ze všech relevantních dokumentů obsažených v korpusu. Rovnice 5.1 a 5.2 zobrazují vzorec pro získání *přesnosti* a *úplnosti*.

$$přesnost = \frac{|\{\text{relevantní dokumenty}\} \cap \{\text{získané dokumenty}\}|}{|\{\text{získané dokumenty}\}|} \tag{5.1}$$

$$úplnost = \frac{|\{\text{relevantní dokumenty}\} \cap \{\text{získané dokumenty}\}|}{|\{\text{relevantní dokumenty}\}|} \tag{5.2}$$

F-míra, jinak také F_1 *score* je komplexnější a kombinuje obě předchozí zmíněné míry dohromady, konkrétně se jedná o jejich harmonický průměr. Vzorec pro její výpočet je uveden v rovnici 5.3.

$$F = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}} \tag{5.3}$$

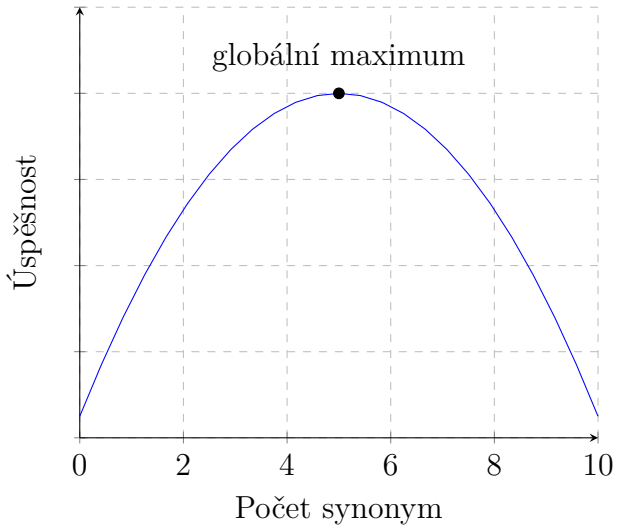
Dále použijeme *střední průměrnou přesnost* (anglicky *mean average precision* zkráceně *MAP*). Tuto metriku jsme využili pro rychlé a jednoduché ověření kvality u jednotlivých měření, jelikož ji program `trec_eval` přiřazuje vysokou prioritu a v jeho výstupu je uvedena mezi prvními.

5.2 Porozumění dotazu

Za porozumění významu dotazu v tomto případě považujeme nalezení významu jednotlivých slov z dotazu a následné obohacení vyhledávaného dotazu o slova s podobným významem získaná z modelů *Word2vec* a *FastText*. Doplněním těchto slov předpokládáme, že bude výsledek dotazu obsahovat takové formulace, které by v případě vyhledávání pomocí klíčových slov nebylo možno najít. Dále předpokládáme, že se výrazně zlepší úplnost dotazu bez významného snížení přesnosti.

5.3 Sestavení hypotézy

Prvním krokem předcházejícím samotné sémantické reprezentaci textu je sestavení hypotézy, s jejíž pomocí dojde ke zlepšení relevantnosti vyhledávání v námi zvoleném full-textovém vyhledávači *Apache Solr*, který byl podrobně představen v předchozí kapitole.



Obrázek 5.1: Navržená hypotéza

Naším cílem je rozšíření dotazu o synonyma získaná pomocí modelů probraných v kapitole 2.

Na obrázku 5.1 můžeme vidět graf[1] zobrazující závislost počtu synonym na úspěšnosti vyhledávání. Naším cílem je nalézt bod označený jako globální maximum nebo se k tomuto bodu co nejvíce přiblížit. Primární snahou je

[1]Funkce zobrazená na grafu neodráží skutečné chování vyhledávání, nýbrž slouží pouze k demonstraci námi navrhovaného řešení.

především nalezení co možná nejvyššího možného počtu relevantních dokumentů, jinými slovy, zvýšení *úplnosti* vyhledávání.

5.4 Příklad sémantické reprezentace

Příklady synonym získaných pomocí modelu *Word2vec* si můžeme prohlédnout v tabulce 5.1. V tomto případě se jednalo o model trénovaný pouze na datech získaných z internetové encyklopedie *Wikipedia*.

Tabulka 5.1: Synonyma *Word2vec*

auto		**král**		**fotbal**		**divadlo**	
slovo	$cos(\theta)$	slovo	$cos(\theta)$	slovo	$cos(\theta)$	slovo	$cos(\theta)$
vozidlo	0.608	*panovník*	0.700	*basketbal*	0.741	*divadle*	0.607
auta	0.572	*císař*	0.643	*florbal*	0.641	*hadivadlo*	0.558
automobil	0.565	*králem*	0.629	*hokej*	0.637	*divadélko*	0.557
kupé	0.562	*trůn*	0.593	*volejbal*	0.625	*divadla*	0.556
suv	0.560	*vévoda*	0.575	*házená*	0.611	*činohra*	0.540
superb	0.530	*vzdorokrál*	0.573	*futsal*	0.611	*divadlem*	0.524
automobily	0.526	*kníže*	0.573	*hokejbal*	0.597	*divadelní*	0.511
kamion	0.524	*vládce*	0.572	*tenis*	0.585	*činoherní*	0.507
audi	0.516	*princ*	0.571	*ragby*	0.562	*molière*	0.506
yeti	0.512	*krále*	0.560	*baseball*	0.541	*rokoko*	0.503

Pro slovo *auto* byla mezi prvními nalezena synonyma *vozidlo* a *automobil*. Konkrétně slova *auto* a *vozidlo* sdílejí nejvyšší kosinovou podobnost. Obdobně pro výraz *král* byla nalezena synonyma *panovník* a *císař*. Výsledky jsou tedy na první pohled velmi dobré.

Při bližším pohledu si však povšimneme potenciálního problému. Především u synonym nalezených pro výraz *fotbal* se objevují různé sporty, avšak v případě, že je vyhledávaný dotaz formulován jako „fotbalový zápas“, velmi pravděpodobně výsledky pro dotazy „hokejový zápas“ nebo „tenisový zápas“ pro nás nebudou mít relevantní hodnotu. Nejedná se tedy o přesná synonyma, která by bylo možné v kontextu volně zaměnit, aniž by došlo ke změně významu sdělení.

Je zřejmé, že synonyma získaná z modelu vytvořeného pomocí metody *FastText*, která jsou uvedena v tabulce 5.2, se na první pohled liší od synonym získaných pomocí *Word2vec*. Až na několik výjimek se jedná o slova zcela odlišná. Model *Word2vec* preferuje především slova s podobným či stejným významem, respektive klade důraz na sémantiku slova. Naproti tomu pro model *FastText* je důležitější samotná skladba slova a zaměřuje se spíše

Tabulka 5.2: Synonyma *FastText*

auto		**král**		**fotbal**		**divadlo**	
slovo	$cos(\theta)$	slovo	$cos(\theta)$	slovo	$cos(\theta)$	slovo	$cos(\theta)$
autogyro	0.821	*exkrál*	0.899	*db_fotbal*	0.975	*idivadlo*	0.969
automoto	0.804	*kukrál*	0.855	*efotbal*	0.971	*pidivadlo*	0.956
autozug	0.801	*králik*	0.780	*csfotbal*	0.964	*hadivadlo*	0.923
sauto	0.772	*vicekrál*	0.774	*sigmafotbal*	0.899	*nedivadlo*	0.896
autotaxi	0.763	*vzdorokrál*	0.771	*fotbalportal*	0.882	*divadlom*	0.894
autosurf	0.761	*velkokrál*	0.767	*fotball*	0.871	*kladivadlo*	0.862
autogiro	0.752	*velekrál*	0.761	*eurofotbal*	0.865	*divadýlko*	0.860
autozam	0.747	*královstwí*	0.737	*fotbalnews*	0.864	*profidivadlo*	0.846
flauto	0.745	*královstwj*	0.730	*nemeckyfotbal*	0.842	*divadlopolarka*	0.831
autovía	0.738	*králi*	0.692	*fotbalunas*	0.838	*divadly*	0.823

na slova s totožným či velmi podobným kořenem. Tato skutečnost pravděpodobně nastává vzhledem k metodě n-gramů, podrobněji zmíněné v kapitole 2, kterou *FastText* používá při tvorbě slovních vektorů. Zároveň jsou hodnoty kosinové vzdálenosti znatelně vyšší u *FastText* než u *Word2vec*, a to patrně ze stejného důvodu. Je pravděpodobné, že výsledky experimentů z kapitoly 6 se z tohoto důvodu budou pro oba modely lišit. V následující části se zaměříme na tvorbu a výslednou formu rozšířeného dotazu.

5.5 Rozšířený dotaz

Rozšířený dotaz vznikne přidáním synonym získaných z natrénovaného modelu k textu dotazu. Uvažujme například, že se uživatel pokouší vyhledat následující dotaz: „Nehody v zaměstnání“. Rozšířený dotaz sestavíme přidáním slov *havárie* a *neštěstí* ke slovu *nehody*. Do druhé části dotazu, konkrétně ke slovu *zaměstnání* přidáme výrazy *povolání* a *podnikání*. Je pravděpodobné, že tímto způsobem rozšířený dotaz dokáže nalézt více článků týkajících se daného tématu. Získané články v zásadě nemusí obsahovat slova z původního uživatelského dotazu, zároveň se však jeho význam doplněním dalších slov téměř nezměnil.

Ovšem i zde může nastat problém se záměnou významu slov. Například pro slova *havárie* nebo *neštěstí* přidaná do rozšířeného dotazu může vyhledávač vrátit články týkající se například *automobilové havárie* nebo *leteckého neštěstí*, které pro uživatele nejsou relevantní. Této skutečnosti se pravděpodobně nevyhneme, avšak právě pro tento případ používá full-textový vyhledávač *Apache Solr* interní ohodnocení relevance nalezených dokumentů, takzvané bodování. Díky tomu mají dokumenty nebo konkrétně v našem pří-

padě články obsahující více vyhledávaných termínů vyšší skóre. Dokumenty s vyšším bodovým ohodnocením jsou intuitivně uváděny přednostně, takže je uživatel uvidí jako první.

6 Experimenty

Účelem této kapitoly bude představit experimenty provedené s natrénovanými modely *Word2vec* a *FastText* s užitím rozličných vstupních datových korpusů. Nejdříve se zaměříme na základní experimenty. Dále uvedeme experimenty s lineární kombinací modelů.

6.1 Volba sémantické reprezentace

Cílem následujících experimentů je volba nejlepší sémantické reprezentace pro danou úlohu. Jednotlivé experimenty se od sebe liší volbou datového korpusu použitého pro natrénování neuronové sítě. V prvním případě byla jako vstupní data použita internetová encyklopedie *Wikipedia*. Jednalo se o verzi z října 2018 sestávající téměř ze 400 tisíc článků a 679 tisíc jedinečných slov. Pro druhý experiment jsme pro natrénování využili pouze samotné novinové články obsažené v datové kolekci **CLEF**. Korpus obsahoval celkem 81 735 článků ze dvou různých periodik. Třetí a současně poslední experiment spojil oba předchozí přístupy dohromady. Výsledkem byl model obsahující vektory pro 723 tisíc slov.

Model trénovaný na *Wikipedii* byl testován celkem na 43 tématech. Zbývajících 7 témat nebylo do experimentů zařazeno z důvodu chybějících synonym, což znemožnilo sestavení rozšířeného dotazu. Značná část nezařazených témat obsahovala vlastní jména, jež nebyla součástí datového korpusu, který byl využit ve fázi tréninku neuronových sítí. V druhém případě se projevila menší velikost vstupního korpusu pro trénink, který umožnil provádět experimenty pouze s 38 tématy. Nejlepší výsledek byl dosažen v posledním případě, kdy bylo testováno nejvíce témat. Celkem se jednalo o 47 témat. Vysoký počet byl umožněn kombinací relativně velkého korpusu *Wikipedie* a speciálních výrazů vyskytujících se pouze v kolekci článků.

6.1.1 Trénování neuronových sítí

Natrénování modelů *Word2vec* a *FastText* bylo provedeno v prostředí programovacího jazyka *Python 3* za pomoci velmi efektivní knihovny *Gensim* (na výkonných strojích byl celý proces trénování dokončen v řádu hodin) vysoce optimalizované pro výpočet na vícejaderných procesorech. Nastavení parametrů bylo v maximální možné míře zachováno stejné pro oba modely, respektive dimenze slovních vektorů byla nastavena na hodnotu 300 a po-

čet trénovacích epoch byl stanoven na 5[1]. Výstupem byl soubor obsahují vypočtené slovní vektory v binární či textové podobě.

6.2 Kombinace modelu *Word2vec* a *FastText*

Po dokončení výše uvedených pokusů jsme se zaměřili na pokročilejší experimenty. Cílem bylo pokusit se využít „silnější“ stránky obou modelů pro další vylepšení výsledků. Z tohoto důvodu jsme zvolili pro další experimenty lineární kombinaci modelů *Word2Vec* a *FastText*.

Než si však představíme navrhované řešení, připomeňme si nejdříve jeden ze základních pojmů lineární algebry, kterým je lineární kombinace. Lineární kombinací vektorů $\boldsymbol{a_1}, \boldsymbol{a_2}, \ldots, \boldsymbol{a_n}$ $(n \in \mathbb{N})$ nazýváme vektor

$$\boldsymbol{a} = c_1\boldsymbol{a_1} + c_2\boldsymbol{a_2} + \ldots + c_n\boldsymbol{a_n},$$

kde c_i $(i = 1, \ldots, n)$ jsou reálná čísla, zvaná *koeficienty lineární kombinace* [1].

Výše uvedený vzorec je obecně platný pro n vektorů. V našem případě se však jedná o lineární kombinaci pouze dvou vektorů. Pro naše potřeby si tedy tento vzorec modifikujeme. Výsledný vektor má poté podobu

$$\boldsymbol{a} = c_1 * \cos(Word2vec) + (1 - c_1) * \cos(FastText), \tag{6.1}$$

kde $w \in (0, 1)$. Rovnice 6.1 již zobrazuje praktické využití lineární kombinace obou modelů. Parametr c_1 umožňuje zvýšit váhu jednoho z vybraných modelů, popřípadě mu ji snížit. Za pomoci těchto experimentů se pokusíme nalézt ideální hodnotu koeficientu c_1. Ideální hodnota označuje stav, při kterém dojde k maximalizaci hodnot sledovaných metrik (detailně probraných v kapitole 5.1) během provádění experimentů.

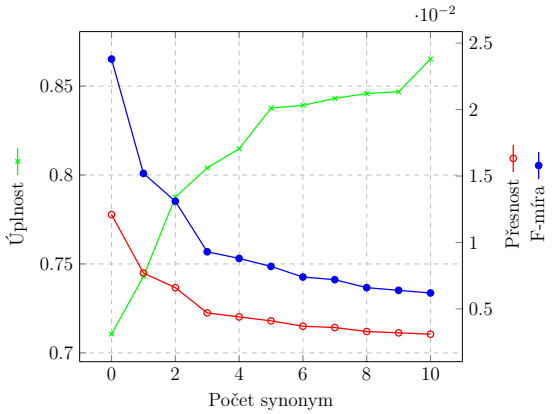
6.3 Dosažené výsledky

Nyní přejdeme k samotnému vyhodnocení výsledků získaných z experimentů zmíněných a podrobně probraných v kapitole 6. Nejdříve se zaměříme na experimenty provedené s oběma zmíněnými modely *Word2vec* a *FastText* natrénovanými na samotné *Wikipedii*.

[1]Vyšší hodnoty znatelně prodloužily dobu nutnou k natrénování, avšak výsledná synonyma byla téměř totožná pro nastavení s námi zvolenými hodnotami.

6.3.1 Modely trénované na *Wikipedii*

Na obrázku 6.1 můžeme pozorovat závislost monitorovaných metrik na počtu synonym přidaných do rozšířeného dotazu. Konkrétně se jedná o výsledky pro 0 až 10 synonym. Všechny naměřené hodnoty jsou současně zaneseny do tabulky 6.1. Graf je rozdělen na dvě osy y. Na levé ose y můžeme sledovat vývoj hodnot pro *úplnost*, zatímco pravá zobrazuje hodnoty pro *přesnost* a *F-míru*.



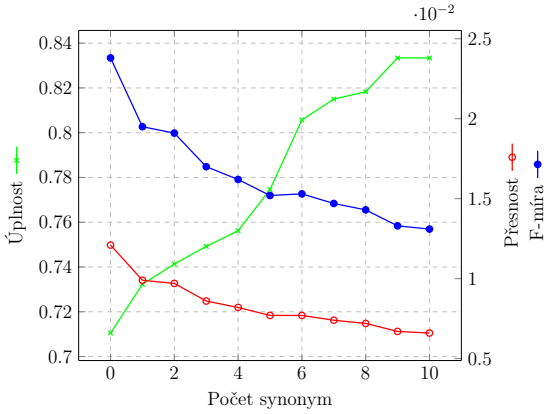
Obrázek 6.1: *Word2vec* trénovaný na *Wikipedii*

Tabulka 6.1: Výsledky pro *Word2vec* trénovaný na *Wikipedii*

#	Precision	Recall	F1	MAP
0	**0.0121**	0.7105	**0.0238**	**0.0118**
1	0.0077	0.7430	0.0152	0.0074
2	0.0066	0.7876	0.0131	0.0068
3	0.0047	0.8040	0.0093	0.0064
4	0.0044	0.8148	0.0088	0.0060
5	0.0041	0.8376	0.0082	0.0060
6	0.0037	0.8392	0.0074	0.0054
7	0.0036	0.8431	0.0072	0.0052
8	0.0033	0.8458	0.0066	0.0044
9	0.0032	0.8468	0.0064	0.0042
10	0.0031	**0.8652**	0.0062	0.0040

Z funkcí zobrazených v grafu 6.1 a hodnot tabulky 6.1 je evidentní, že se *úplnost* zvýšila ze 71% až na více než 86%, konkrétně na hodnotu $\mathbf{86,52}\%$. Jedná se tedy o zlepšení o znatelných 15%. Rozšířený dotaz nám tedy v tomto případě pomohl vyhledat mnohem více **relevantních** článků než původní dotaz. Přesněji uvedeno, z celkového počtu 647 relevantních dokumentů indexovaných pomocí vyhledávače *Apache Solr* dokázal rozšířený

dotaz nalézt **537** témat oproti hodnotě 418 v případě původního dotazu. Na druhé straně došlo k téměř nevyhnutelnému poklesu hodnot pro *přesnost* a *F-míru*. U obou metrik jsme mohli zaznamenat snížení okolo 75% oproti původním hodnotám. Tento efekt nastal z důvodu navrácení článků, které se nevážou k vyhledávanému tématu, avšak obsahují slova z rozšířeného dotazu. V kapitole 6.5 se však blíže zaměříme také na témata, u kterých došlo naopak ke zvýšení všech tří hlavních sledovaných metrik, *přesnost* a *F-míru* nevyjímaje. Nyní se podíváme na výsledky v případě užití modelu *FastText*.



Obrázek 6.2: *FastText* trénovaný na *Wikipedii*

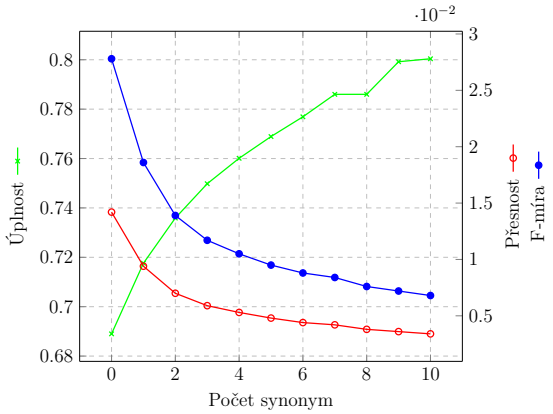
Tabulka 6.2: Výsledky pro *FastText* trénovaný na *Wikipedii*

#	Precision	Recall	F1	MAP
0	**0.0119**	0.6953	**0.0234**	**0.0117**
1	0.0100	0.7215	0.0197	0.0113
2	0.0098	0.7316	0.0193	0.0111
3	0.0087	0.7419	0.0172	0.0098
4	0.0083	0.7477	0.0164	0.0093
5	0.0078	0.7660	0.0154	0.0092
6	0.0078	0.7971	0.0154	0.0095
7	0.0075	0.8065	0.0149	0.0088
8	0.0073	0.8098	0.0145	0.0087
9	0.0069	0.8282	0.0137	0.0084
10	0.0067	**0.8282**	0.0133	0.0078

Z grafu 6.2 je patrné, že stejně jako v případě pro model *Word2Vec* došlo i u modelu *FastText* k výraznému zvýšení hodnot pro *úplnost*. Zároveň se uskutečnil drobný pokles hodnot pro *přesnost* a *F-míru* podobně jako u *Word2Vec*. *Úplnost* se zde zastavila na hodnotě **83, 34**%. Celkem bylo s použitím synonym z modelu trénovaného pomocí *FastText* nalezeno **506** relevantních témat.

Ve výsledku tedy rozšířený dotaz využívající model *Word2Vec* dosáhl o více než 3% větší *úplnost* vyhledávání než *FastText*. Druhý zmiňovaný však dosahoval vyšších hodnot pro *přesnost*, což nás vedlo k využití kombinace obou modelů pro další potenciální vylepšení doposud získaných výsledků. Následně se zaměříme na výsledky obou modelů trénovaných pouze na datech z datové kolekce CLEF.

6.3.2 Modely *Word2vec* a *FastText* trénované na datech z kolekce CLEF



Obrázek 6.3: *Word2vec* trénovaný pouze na kolekci **CLEF**

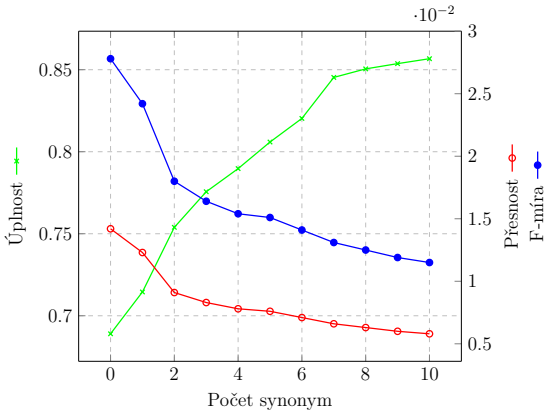
Graf 6.3 zobrazuje výsledky získané pro *Word2vec* trénovaný pouze na datové kolekci **CLEF**. Všechny hodnoty jsou opět zanesené do tabulky 6.3. Z úplného počtu 583 relevantních témat (o 64 méně než v předchozím případě) bylo s pomocí tohoto modelu nalezeno **443** z nich, což představuje *úplnost* přesahující 80%. I zde došlo k určitému zlepšení, nicméně se nejedná o tak výrazný nárůst jako v případě, kdy byla jako trénovací množina využita kompletní *Wikipedia*. K tomuto efektu došlo patrně z důvodů menší trénovací množiny, která neobsahovala dostatečný počet slovních výrazů. Přesnost stejně jako v předchozím případě poklesla přibližně o 75%.

Oproti modelu *Word2vec* dosáhl *FastText* (viz graf 6.4 a tabulka 6.4) ve výsledku o celých 5% vyšší hodnoty pro *úplnost*. Bylo nalezeno přesně **489** témat. V případě nevelkého vstupního korpusu dokáže tedy pravděpodobně lépe pracovat *FastText*.

Nastal i pokles *přesnosti*, obdobně jako v předchozím v případě. Jednalo se o snížení přibližně o 60%.

Tabulka 6.3: Výsledky pro *Word2vec* trénovaný na kolekci **CLEF**

#	Precision	Recall	F1	MAP
0	**0.0142**	**0.6890**	**0.0278**	**0.0136**
1	0.0094	0.7198	0.0186	0.0096
2	0.0070	0.7381	0.0139	0.0083
3	0.0059	0.7520	0.0117	0.0070
4	0.0053	0.7638	0.0105	0.0068
5	0.0048	0.7726	0.0095	0.0061
6	0.0044	0.7829	0.0088	0.0058
7	0.0042	0.7941	0.0084	0.0053
8	0.0038	0.7941	0.0076	0.0050
9	0.0035	0.8058	0.0070	0.0044
10	0.0034	**0.8070**	0.0068	0.0043



Obrázek 6.4: *FastText* trénovaný pouze na kolekci **CLEF**

6.3.3 Modely trénované na *Wikipedii* i kolekci CLEF

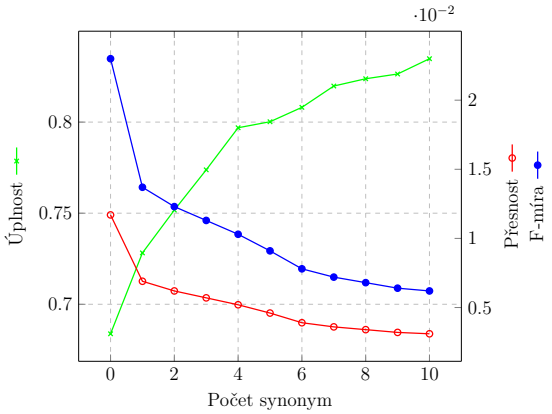
V této kapitole se zaměříme na výsledky experimentů, které byly prováděny s modely trénovanými na kombinaci dat z *Wikipedie* a současně s články z datového korpusu **CLEF**. Byl použit největší datový korpus pro učení neuronové sítě jak *Word2vec*, tak i *FastText*. Datový korpus dohromady obsahoval více než 723 tisíc různých slov.

V tomto případě dosáhl model *úplnosti* 83, 48%. Jedná se tedy o lepší výsledek než při použití pouze **CLEF** jako datového korpusu, avšak nižší než v případě *Wikipedie*. Bylo nalezeno dohromady **568** relevantních témat z celkového počtu **691**.

Oproti tomu *FastText* zaznamenal výrazný pokles *úplnosti* oproti předchozím případům a její hodnota se nedostala přes 80%, respektive bylo nalezeno jen **537** relevantních témat. *Word2vec* tedy vyhledal o 31 témat více.

Tabulka 6.4: Výsledky pro *FastText* trénovaný na kolekci **CLEF**

#	Precision	Recall	F1	MAP
0	**0.0142**	0.6890	**0.0278**	0.0136
1	0.0123	0.7145	0.0242	0.0117
2	0.0091	0.7539	0.0180	0.0093
3	0.0083	0.7756	0.0164	0.0091
4	0.0078	0.7897	0.0154	0.0091
5	0.0076	0.8059	0.0151	0.0089
6	0.0071	0.8203	0.0141	0.0085
7	0.0066	0.8453	0.0131	0.0082
8	0.0063	0.8505	0.0125	0.0080
9	0.0060	0.8537	0.0119	0.0075
10	0.0058	**0.8567**	0.0115	**0.0075**



Obrázek 6.5: *Word2vec* trénovaný současně na *Wikipedii* i kolekci **CLEF**

Přesto si model *FastText* zachoval lepší hodnoty pro *přesnost* a *F-míru* než *Word2vec* v předchozích případech.

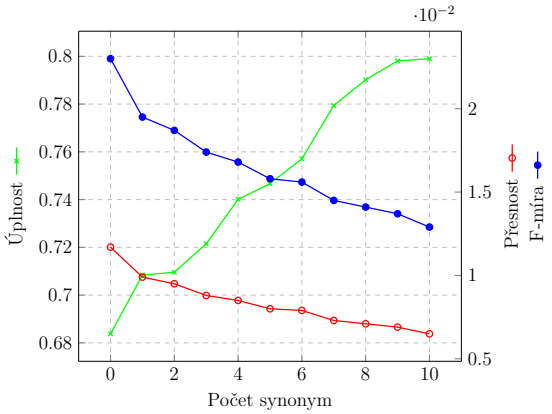
6.3.4 Shrnutí naměřených výsledků

Z naměřených hodnot vyplývá, že oba modely *Word2vec* i *FastText* dokázaly zvýšit úplnost vyhledávání téměř o 15% . Při využití modelu *Word2vec* dosahovala *úplnost* v určitých případech téměř 87%. Podařilo se prokázat, že s využitím obou modelů k sestavení rozšířeného dotazu lze nalézt více relevantních témat, než tomu bylo v případě běžného dotazu.

Ve srovnání s modelem *Word2vec* dosahoval *FastText* vyšších hodnot pro *přesnost* a *F-míru*. Tato skutečnost nastala patrně v důsledku využití n-gramů daným modelem. Z tohoto důvodu došlo k nalezení slov, která jsou tvarově blíže (například *automobil* – *automobilový*) než slova, jež našel

Tabulka 6.5: Výsledky pro *Word2vec* trénovaný na *Wikipedii* i kolekci **CLEF**

#	Precision	Recall	F1	MAP
0	**0.0117**	0.6838	**0.0230**	**0.0112**
1	0.0069	0.7282	0.0137	0.0067
2	0.0062	0.7517	0.0123	0.0065
3	0.0057	0.7738	0.0113	0.0068
4	0.0052	0.7969	0.0103	0.0066
5	0.0046	0.8002	0.0091	0.0062
6	0.0039	0.8081	0.0078	0.0049
7	0.0036	0.8198	0.0072	0.0047
8	0.0034	0.8238	0.0068	0.0043
9	0.0032	0.8264	0.0064	0.0041
10	0.0031	**0.8348**	0.0062	0.0040



Obrázek 6.6: *FastText* trénovaný současně na *Wikipedii* i kolekci **CLEF**

model *Word2vec*. Ze stejného důvodu však *FastText* nalezl méně relevantních dokumentů.

6.4 Výsledky kombinace modelu *Word2vec* a *FastText*

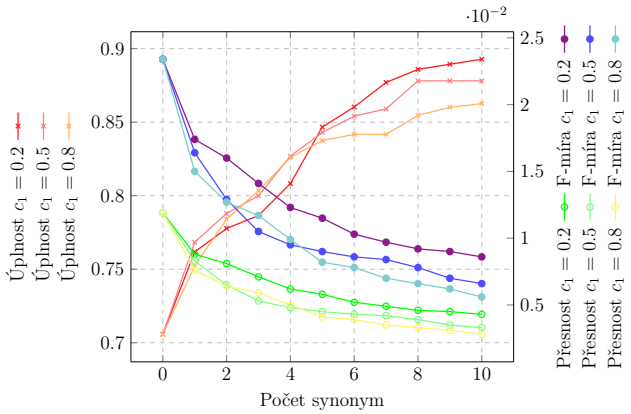
V této části prostudujeme výsledky pro experimenty s lineární kombinací modelů *Word2vec* a *FastText* probrané v kapitole 6.2. Projdeme výsledky pro různá nastavení parametru c_1, konkrétně pro hodnoty 0.2, 0.5 a 0.8. Vyšší hodnoty parametru c_1 znamenají vyšší prioritu pro model *Word2vec* a naopak.

V grafu 6.7 můžeme vidět vývoj hodnot pro *úplnost*, *přesnost* a *F-míru* s různou volbou velikosti parametru c_1. Lze si povšimnout zvýšení všech

Tabulka 6.6: Výsledky pro *FastText* trénovaný na *Wikipedii* i kolekci **CLEF**

#	Precision	Recall	F1	MAP
0	**0.0117**	0.6838	**0.0230**	**0.0112**
1	0.0099	0.7084	0.0195	0.0109
2	0.0095	0.7096	0.0187	0.0105
3	0.0088	0.7215	0.0174	0.0101
4	0.0085	0.7401	0.0168	0.0096
5	0.0080	0.7467	0.0158	0.0087
6	0.0079	0.7571	0.0156	0.0088
7	0.0073	0.7794	0.0145	0.0087
8	0.0071	0.7902	0.0141	0.0085
9	0.0069	0.7981	0.0137	0.0084
10	0.0065	**0.7990**	0.0129	0.0078

sledovaných metrik při postupném snižování priority *Word2vec*. Mírně překvapivý fakt je zvýšení hodnot *úplnosti* při volbě parametru $c_1 = 0.2$, tedy upřednostnění modelu *FastText*. V tomto případě dosáhla *úplnost* dokonce více než 89%, což je lepší než dosavadní nejlepší výsledek pro model *Word2vec* trénovaný na samotné *Wikipedii*. Při zvolení $c_1 = 0.2$ tedy zřejmě došlo k nalezení ideálního poměru.



Obrázek 6.7: Vývoj metrik při různém nastavení parametru c_1

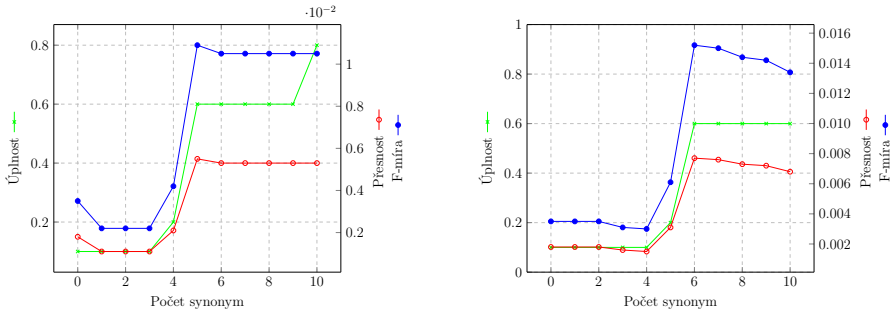
S využitím lineární kombinace uvedených modelů lze tedy úspěšně dále vylepšovat *úplnost* vyhledávání. Ani v tomto případě však nedošlo ke zlepšení hodnot pro *přesnost* a *F-míru*.

6.5 Výsledky pro jednotlivá témata

V závěru této kapitoly se podrobněji zaměříme na výsledky, které byly získány pro vybraná témata. Vzhledem k jejich relativně vysokému počtu však zmíníme pouze 2 témata. Konkrétně se bude jednat o následující témata *425-AH – Ohrožené druhy* a *449-AH – Občanské války v Africe*. Zvolená témata dosáhla během experimentů mimořádných výsledků.

Stejně jako v případě předchozích experimentů uvedeme a detailně analyzujeme všechny sledované metriky, navíc také ke každému z analyzovaných témat budou uvedena slova použitá v rozšířeném dotazu, která dokázala zlepšit výsledky.

6.5.1 Téma *Ohrožené druhy*



(a) *Word2Vec* (b) *FastText*

Obrázek 6.8: Výsledky pro téma *425-AH*

Na obrázku 6.8 můžeme vidět výsledky pro téma *425-AH Ohrožené druhy* současně pro oba modely *Word2vec* i *FastText*. Na první pohled je zcela zřejmé, že došlo k velmi výraznému nárůstu hodnot všech sledovaných metrik. U obou modelů bylo možné zaznamenat zvýšení hodnot nejen pro *úplnost*, ale také pro *přesnost* a samotnou *F-míru*. Nejvyšší nárůst je možné sledovat u počtu 5 synonym přidaných do rozšířeného dotazu v případě *Word2vec*, respektive 6 synonym u modelu *FastText*. Na rozdíl od *FastText*, který dokázal nalézt jen **6** z celkového počtu **10** relevantních článků, rozšířené dotazy používající model *Word2vec* vrátily **8** relevantních dokumentů (konkrétně pro případ 10 synonym). Na druhé straně *FastText* dosáhl vyšších hodnot pro *přesnost* a *F-míru*. Došlo tedy k více než čtyřnásobnému nárůstu hodnot těchto metrik (pouze trojnásobnému u modelu *Word2vec)*. Výsledné hodnoty experimentů jsou uvedeny v tabulkách 6.7 a 6.8.

V obou případech bylo do rozšířeného dotazu přidáno slovo *ohrožených*, které výrazně pomohlo najít více relevantních článků k danému tématu.

Tabulka 6.7: Výsledky pro téma *AH-425* model *Word2vec*

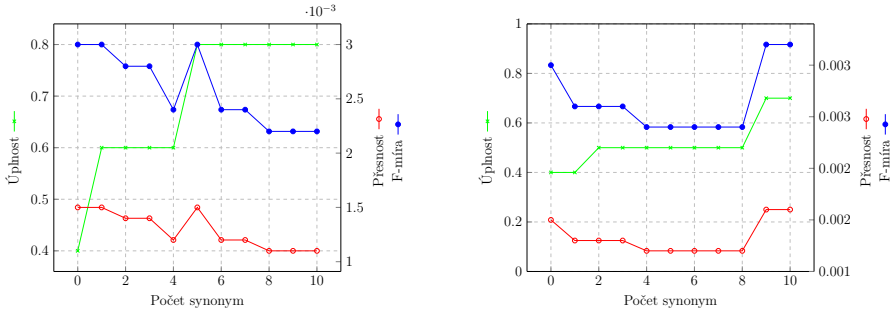
#	Precision	Recall	F1	MAP	Rel	RelRet	Ret
0	0.0018	0.1000	0.0035	0.0002	10	1	566
1	0.0011	0.1000	0.0022	0.0001	10	1	904
2	0.0011	0.1000	0.0022	0.0001	10	1	915
3	0.0011	0.1000	0.0022	0.0001	10	1	925
4	0.0021	0.2000	0.0042	0.0004	10	2	972
5	**0.0055**	0.6000	**0.0109**	**0.0092**	10	6	1099
6	0.0053	0.6000	0.0105	0.0091	10	6	1123
7	0.0053	0.6000	0.0105	0.0091	10	6	1129
8	0.0053	0.6000	0.0105	0.0091	10	6	1137
9	0.0053	0.6000	0.0105	0.0091	10	6	1137
10	0.0053	**0.8000**	0.0105	0.0084	10	**8**	1517

Tabulka 6.8: Výsledky pro téma *AH-425* model *FastText*

#	Precision	Recall	F1	MAP	Rel	RelRet	Ret
0	0.0018	0.1000	0.0035	0.0002	10	1	566
1	0.0018	0.1000	0.0035	0.0001	10	1	570
2	0.0018	0.1000	0.0035	0.0001	10	1	571
3	0.0016	0.1000	0.0031	0.0001	10	1	644
4	0.0015	0.1000	0.0030	0.0004	10	1	647
5	0.0031	0.2000	0.0061	**0.0092**	10	2	652
6	**0.0077**	**0.6000**	**0.0152**	0.0091	10	**6**	780
7	0.0076	0.6000	0.0150	0.0091	10	6	794
8	0.0073	0.6000	0.0144	0.0091	10	6	827
9	0.0072	0.6000	0.0142	0.0091	10	6	834
10	0.0068	0.6000	0.0134	0.0084	10	6	888

6.5.2 Téma *Občanské války v Africe*

U tématu *AH-429 – Občanské války v Africe* byly naměřené hodnoty maximální pro 5 synonym u modelu *Word2vec* a 9 synonym v případě *FastText*. Z grafu 6.9 je patrné, že stejně jako v předchozím případě i zde dosáhl *Word2vec* vyšší *úplnosti*. Z **10** relevantních témat nalezl **8** a *FastText* jen o jedno méně. Do rozšířených dotazů byla přidána slova *válce*, *válkou* a *občanskou*, která pomohla zvýšit hodnoty *úplnosti*. Souhrnné výsledky experimentů pro téma *449-AH* jsou uvedeny v tabulkách 6.9 a 6.10.



(a) *Word2Vec* (b) *FastText*

Obrázek 6.9: Výsledky pro téma *449-AH*

Tabulka 6.9: Výsledky pro téma *AH-449* model *Word2vec*

#	Precision	Recall	F1	MAP	Rel	RelRet	Ret
0	0.0015	0.4000	0.0030	0.0005	10	4	2646
1	0.0015	0.6000	0.0030	0.0007	10	6	3180
2	0.0014	0.6000	0.0028	0.0006	10	6	3980
3	0.0014	0.6000	0.0028	0.0006	10	6	3998
4	0.0012	0.6000	0.0024	0.0006	10	6	4173
5	**0.0015**	**0.8000**	**0.0030**	**0.0011**	10	**8**	4189
6	0.0012	0.8000	0.0024	0.0009	10	8	4192
7	0.0012	0.8000	0.0024	0.0009	10	8	4202
8	0.0011	0.8000	0.0022	0.0009	10	8	4259
9	0.0011	0.8000	0.0022	0.0008	10	8	4424
10	0.0011	0.8000	0.0022	0.0008	10	8	4430

Tabulka 6.10: Výsledky pro téma *AH-449* model *FastText*

#	Precision	Recall	F1	MAP	Rel	RelRet	Ret
0	0.0015	0.4000	0.0030	0.0002	10	4	2646
1	0.0013	0.4000	0.0026	0.0001	10	4	3922
2	0.0013	0.5000	0.0026	0.0001	10	5	4216
3	0.0013	0.5000	0.0026	0.0001	10	5	4441
4	0.0012	0.5000	0.0024	0.0004	10	5	4916
5	0.0012	0.5000	0.0024	**0.0092**	10	5	5303
6	0.0012	0.5000	0.0024	0.0091	10	5	6571
7	0.0012	0.5000	0.0024	0.0091	10	5	6667
8	0.0012	0.5000	0.0024	0.0091	10	5	7033
9	**0.0016**	**0.7000**	**0.0032**	0.0091	10	**7**	7217
10	0.0016	0.7000	0.0032	0.0084	10	7	7343

7 Závěr

V analytické části práce jsme prozkoumali základní metody sémantické reprezentace textu, jejich výhody a nevýhody. V dalším kroku jsme se seznámili s dostupnými datovými kolekcemi a na základě provedené analýzy vybrali kolekci **CLEF** vhodnou pro ověření kvality vyhledávání informací. Současně jsme detailně prostudovali full-textový vyhledávač *Apache Solr*, do kterého jsme zmíněné metody integrovali.

V praktické části bakalářské práce byl popsán návrh a implementace metod pro sémantickou reprezentaci textu. Poté proběhlo ověření jejich funkčnosti na zvolené datové kolekci s následným vyhodnocením naměřených výsledků a pokusem navrhnout možná vylepšení.

S použitím modelů *Word2vec* a *FastText* se ve všech uvedených experimentech podařilo zvýšit hodnoty *úplnosti* vyhledávání pouze s mírným snížením *přesnosti*. K dalšímu navýšení hodnot *přesnosti* by téměř jistě došlo za předpokladu, že bychom použili pro dané výrazy výhradně synonyma.

Dosažené výsledky v rámci testování uvedených metod vytváří prostor pro další varianty využití. Jedním z mnoha možných rozšíření bakalářské práce je použití lepší kombinace modelů. Dále je to také kombinace použití *stemmingu* a odstranění *stop-slov* ze vstupního datového korpusu. Komplexním rozšířením bakalářské práce by bylo natrénování použitých modelů *Word2vec* a *FastText* na vícejazyčném korpusu pro získání výsledků vyhledávání v různých jazycích.

Seznam zkratek

CBOW Continuous Bag-of-Words

CDDS Clinical Decision Support System

CLEF Conference and Labs of the Evaluation Forum

CSV Comma-Separated Values

F1 F_1 score

IDF Inverse Document Frequency

IR Information Retrieval

JSON JavaScript Object Notation

MAP Mean Average Precision

NLP Natural Language Processing

NN Neural Network

REL Relevant Documents

RELRET Relevant Returned Documents

RET Returned Documents

SGML Standard Generalized Markup Language

TF Term Frequency

TREC Text REtrieval Conference

XML Extensible Markup Language

Seznam obrázků

Seznam tabulek

Literatura

[1] Bartsch, H.-J. *Matematické vzorce*. Praha: Mladá fronta, 2000. ISBN 80-204-0607-7.

[2] Firth, J. R. A synopsis of linguistic theory 1930-55. *Studies in Linguistic Analysis*. 1957, s. 1–32.

[3] Gheorghe, R. – Hinman, M. L. – Russo, R. *Elasticsearch in Action*. Manning Publications Co., 2015. ISBN 9781617291623.

[4] Graigner, T. – Potter, T. *Solr in Action*. Shelter Island, 2014. ISBN 9781617291029.

[5] Harris, Z. S. Distributional Structure. *Word*. 1954, 10, 23, s. 146–162. doi: 10.1080/00437956.1954.11659520.

[6] Huang, S. *Word2Vec and FastText Word Embedding with Gensim* [online]. 2018. [cit. 2019-01-18]. Dostupné z: `https://towardsdatascience.com/word-embedding-with-word2vec-and-fasttext-a209c1d3e12c`.

[7] Koetsier, J. *How Google searches 30 trillion web pages, 100 billion times a month* [online]. 2016. [cit. 2018-10-15]. Dostupné z: `https://venturebeat.com/2013/03/01/how-google-searches-30-trillion-web-pages-100-billion-times-a-month/`.

[8] Mařík, V. – Štěpánová, O. – Lažanský, J. *Umělá inteligence (1)*. Praha: Academia, 1993. ISBN 80-200-0496-3.

[9] McCandless, M. – Hatcher, E. – Gospodnetic, O. *Lucene in Action*. Manning Publications Co., second edition, 2010. ISBN 978-1-933988-17-7.

[10] McCulloch, W. S. – Pitts, W. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*. 1943, 5, 4, s. 115–133. doi: 10.1007/BF02478259.

[11] Mikolov, T. et al. Efficient Estimation of Word Representations in Vector Space. *arXiv preprint arXiv:1301.3781*. 2013.

[12] Mikolov, T. et al. Distributed Representations of Words and Phrases and their Compositionality. In *Advances in neural information processing systems*, s. 3111–3119, 2013.

[13] Pennington, J. – Socher, R. – Manning, C. D. GloVe: Global Vectors for Word Representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, s. 1532–1543, 2014.

[14] Ryan, K. J. *Facebook's New Open Source Software Can Learn 1 Billion Words in 10 Minutes* [online]. 2016. [cit. 2019-01-20]. Dostupné z: `https://www.inc.com/kevin-j-ryan/facebook-open-source-fasttext-learns-1-billion-words-in-10-minutes.html`.

[15] Statista. [online]. 2019. [cit. 2019-04-04]. Dostupné z: `https://www.statista.com/statistics/272014/global-social-networks-ranked-by-number-of-users/`.

[16] Sullivan, D. *Google now handles at least 2 trillion searches per year* [online]. 2018. [cit. 2018-10-15]. Dostupné z: `https://searchengineland.com/google-now-handles-2-999-trillion-searches-per-year-250247`.

[17] Tang, D. et al. Learning Sentiment-Specific Word Embedding for Twitter Sentiment Classification. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 1, s. 1555–1565, 2014.

A Uživatelská dokumentace

Na přiloženém DVD jsou umístěny všechny zdrojové kódy rozšíření vyhledávače *Apache Solr* společně s výsledným `jar` souborem. Ve složce *vectors* nalezneme soubor *model_w2v.txt* obsahující slovní vektory trénované pomocí modelu *Word2vec* na *Wikipedii*.

A.1 Konfigurace a spuštění pluginu

Nejdříve je potřeba v kořenové složce jádra vytvořit složku, ideálně pojmenovanou *lib*. Do této složky umístíme dodaný jar soubor. Dále ve složce *lib* vytvoříme podsložku *files*, kam vložíme soubor se slovními vektory.

Současně je také nutné upravit konfigurační soubor *solrcofig.xml* umístěný v jádře ve složce *conf*. Na konec tohoto souboru, konkrétně před koncový tag `</config>` je potřeba přidat následující dva řetězce.

```
<lib dir="/lib" regex=".*\.jar" />
<queryParser name="semanticparser" class="SemanticQParserPlugin">
</queryParser>
```

První řádek odkazuje *Solr* na umístění, odkud lze načíst doplňující knihovny, popřípadě pluginy. V našem případě se jedná o složku *lib*, ve které je umístěn náš jar soubor. Druhý řádek zaregistruje nový *syntaktický analyzátor* (anglicky *parser*) automaticky sestavující rozšířený dotaz pomocí přidání synonym z dodaného modelu ve složce *files*.

Přidáním parametru `deftype=semanticparser` k uživatelskému dotazu docílíme jeho použití pro zpracování textu dotazu. Pokud nechceme manuálně volit náš syntaktický analyzátor při každém dotazu zadáním parametru, je možné nastavit ho jako výchozí přidáním nového tagu.

```
<lst name="defaultst">
       <str name="defType">semanticparser</str>
</lst>
```

Vyhledávač *Solr* bude od této chvíle zpracovávat dotazy pomocí našeho syntaktického analyzátoru. Načtení slovních vektorů ze souboru je vzhledem k velikosti souborů (řádově několik GB) časově náročný proces, z tohoto důvodu první vyhledávání zabere několik minut.

A.2 Obsah doprovodného DVD

Přiložené DVD obsahuje následující složky:

- **text** – obsahuje zdrojové soubory LaTeXa PDF bakalářské práce.
- **vectors** – složka obsahující soubor se slovními vektory, konkrétně model *Word2vec*.
- **application**
  - **doc** – obsahuje dokumentaci projektu.
  - **jar** – složka obsahující výsledný jar soubor s pluginem.
  - **lib** – obsahuje knihovny potřebné k překladu projektu.
  - **out** – obsahuje binární soubory, přeložené pomocí *Java* verze 11.
  - **src** – zde jsou umístěny zdrojové kódy knihovny.
  - **build.xml** – soubor umožňují překlad aplikace pomocí *Ant*.

B Seznam témat

- 10.2452/401-AH – Inflace Eura
- 10.2452/402-AH – Obnovitelné zdroje
- 10.2452/403-AH – Role policisty
- 10.2452/404-AH – Summit NATO a bezpečností opatření
- 10.2452/405-AH – Astma u dětství
- 10.2452/406-AH – Animované filmy
- 10.2452/407-AH – Australský premiér
- 10.2452/408-AH – Klonování lidí
- 10.2452/409-AH – Automobilové bombové útoky v Bali
- 10.2452/410-AH – Porušení programu jaderných zbraní Severní Koreou
- 10.2452/411-AH – Oskar za nejlepší film
- 10.2452/412-AH – Knihy o politicích
- 10.2452/413-AH – Snižování rizika onemocnění cukrovkou
- 10.2452/414-AH – Pivní festivaly
- 10.2452/415-AH – Drogy
- 10.2452/416-AH – Rukojmí v moskevském divadle
- 10.2452/417-AH – Únosy letadel
- 10.2452/418-AH – Prohlášení Bülenta Ecevita
- 10.2452/419-AH – Úložiště jaderného odpadu
- 10.2452/420-AH – Obezita a zdravotní problémy
- 10.2452/421-AH – Olympijské medaile sourozenců Kosteličových
- 10.2452/422-AH – Uzavírání průmyslových a obchodních podniků
- 10.2452/423-AH – Alternativy očkování proti chřipce
- 10.2452/424-AH – Rozvoj internetového bankovnictví
- 10.2452/425-AH – Ohrožené druhy
- 10.2452/426-AH – Protiteroristická opatření
- 10.2452/427-AH – Svědectví proti Miloševičovi
- 10.2452/428-AH – Ekologický turismus
- 10.2452/429-AH – Voda a její zdravotní rizika
- 10.2452/430-AH – Plastické operace
- 10.2452/431-AH – Francouzští prezidenští kandidáti
- 10.2452/432-AH – Prezidentské volby v Zimbabwe
- 10.2452/433-AH – Zneužívání dětí duchovními
- 10.2452/434-AH – Politická nestabilita ve Venezuele
- 10.2452/435-AH – Příčiny znečištění vzduchu
- 10.2452/436-AH – Rozvody významných lidí

- 10.2452/437-AH – Neregulérní audity v Enronu
- 10.2452/438-AH – Výzkum rakoviny
- 10.2452/439-AH – Nehody v zaměstnání
- 10.2452/440-AH – Zimní olympijské hry 2002 a doping
- 10.2452/441-AH – Vesmírní turisté
- 10.2452/442-AH – Pohřeb Královny Matky
- 10.2452/443-AH – Světové rekordy v plavání
- 10.2452/444-AH – Brazilští vítězové fotbalového mistrovství světa
- 10.2452/445-AH – Princ Harry a drogy
- 10.2452/446-AH – Záplavy a ztráty na kulturním dědictví
- 10.2452/447-AH – Pim Fortuynova politika
- 10.2452/448-AH – Nobelovy ceny za chemii
- 10.2452/449-AH – Občanské války v Africe
- 10.2452/450-AH – Neúspěšné pokusy o atentát