

Sem vložte zadání Vaší práce.



**FAKULTA
INFORMAČNÍCH
TECHNOLOGIÍ
ČVUT V PRAZE**

Diplomová práce

Open-source doporučovací systém pro české veřejné zakázky

Bc. Milan Vancl

Katedra aplikované matematiky
Vedoucí práce: Ing. Jaroslav Kuchař, Ph.D.

18. května 2020

Poděkování

Především bych chtěl vyjádřit svůj vděk své rodině, která mě podporovala po celou dobu mého studia od začátku až do konce. Děkuji panu Ing. Jaroslavu Kuchařovi, Ph.D. za vřelou komunikaci při odborném i formálním vedení této práce. Dále děkuji panu Ing. Marku Sušickému za oponenturu a záštitu téma a společnosti Profinit EU, s.r.o. za materiální podporu pro účely této práce. V neposlední řadě děkuji slečně Ing. Lucii Svitákové, panu Ing. Davidu Šenkýřovi a panu Mgr. Martinu Popelovi, Ph.D. za poskytnuté konzultace a panu Michalu Bláhovi děkuji za poskytnutí datasetu.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona, ve znění pozdějších předpisů. V souladu s ust. § 2373 odst. 2 zákona č. 89/2012 Sb., občanský zákoník, ve znění pozdějších předpisů, tímto uděluji nevýhradní oprávnění (licenci) k užití této mojí práce, a to včetně všech počítačových programů, jež jsou její součástí či přílohou a veškeré jejich dokumentace (dále souhrnně jen „Dílo“), a to všem osobám, které si přejí Dílo užít. Tyto osoby jsou oprávněny Dílo užít jakýmkoli způsobem, který nesnižuje hodnotu Díla a za jakýmkoli účelem (včetně užití k výdělečným účelům). Toto oprávnění je časově, teritoriálně i množstevně neomezené. Každá osoba, která využije výše uvedenou licenci, se však zavazuje udělit ke každému dílu, které vznikne (byť jen zčásti) na základě Díla, úpravou Díla, spojením Díla s jiným dílem, zařazením Díla do díla souborného či zpracováním Díla (včetně překladu) licenci alespoň ve výše uvedeném rozsahu a zároveň zpřístupnit zdrojový kód takového díla alespoň srovnatelným způsobem a ve srovnatelném rozsahu, jako je zpřístupněn zdrojový kód Díla.

V Praze dne 18. května 2020

.....

České vysoké učení technické v Praze
Fakulta informačních technologií

© 2020 Milan Vancl. Všechna práva vyhrazena.

Tato práce vznikla jako školní dílo na Českém vysokém učení technickém v Praze, Fakultě informačních technologií. Práce je chráněna právními předpisy a mezinárodními úmluvami o právu autorském a právech souvisejících s právem autorským. K jejímu užití, s výjimkou bezúplatných zákonných licencí a nad rámec oprávnění uvedených v Prohlášení na předchozí straně, je nezbytný souhlas autora.

Odkaz na tuto práci

Vancl, Milan. *Open-source doporučovací systém pro české veřejné zakázky*. Diplomová práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2020.

Abstrakt

V této práci se zabývám návrhem a implementací open-source doporučovacího systému pro české veřejné zakázky. Provádím průzkum a hodnocení současných systémů týkajících se veřejných zakázek. Pro doporučovací algoritmus vybírám vlastnosti zakázek, pro které navrhoji a implementuji jejich získávání či extrakci. Za hlavní vlastnost určuji předmět plnění zakázek, ze kterého extrahuji sémantickou informaci pomocí embeddingu. Doporučovací algoritmus demonstroji v jednoduché webové aplikaci.

Klíčová slova české veřejné zakázky, doporučovací systém, zpracování textu, podobnost textu, text embedding, open-source

Abstract

In this thesis I deal with the design and implementation of an open-source recommender system for Czech public procurement. I conduct research and discussion of current systems related to public procurement. For the recommendation algorithm, I select a set of procurement features for which I design and implement their retrieval and extraction. I determine the main feature as the subject of procurement from which I extract semantic information using embedding. I demonstrate the recommender algorithm with a simple web application.

Keywords Czech public procurement, recommender system, text processing, text similarity, text embedding, open-source

Obsah

Úvod	1
1 Cíl práce	3
2 Stávající řešení	5
2.1 Shrnutí	7
3 Data a doména	9
3.1 Dokumenty veřejných zakázek	9
3.2 Kategorie produktů a služeb	10
3.3 Katalog produktů a služeb	11
3.4 Data o lokacích	12
3.5 Data o předmětu podnikání zadavatelů	12
4 Extrakce vlastností zakázek	13
4.1 Klasifikace dokumentů	13
4.1.1 Klasifikace hierarchických struktur	14
4.1.2 Multi-label klasifikace	16
4.1.3 Analýza CPV kódů v datech	17
4.1.4 Extrakce CPV kódů	20
4.2 Embedding dokumentů	20
4.2.1 Embedding	20
4.2.2 Klasické metody	21
4.2.2.1 Bag-of-words	21
4.2.2.2 Bag-of-n-grams	22
4.2.2.3 tf-idf	22
4.2.3 Modelování témat	22
4.2.3.1 Latent Dirichlet allocation	22
4.2.4 Metody učení bez učitele	23
4.2.4.1 word2vec	23

4.2.4.2	n-gram embeddingy	24
4.2.4.3	Agregace embeddingů slov	24
4.2.4.4	doc2vec	24
4.2.4.5	GloVe	25
4.2.4.6	Skip-thought vektory	25
4.2.4.7	FastSent	25
4.2.4.8	sent2vec	26
4.2.4.9	Quick-thought vektory	26
4.2.4.10	Word Mover's Embedding	26
4.2.4.11	ELMo	28
4.2.4.12	ULMFiT	28
4.2.4.13	Transformers	28
4.2.4.14	USE	29
4.2.4.15	GPT	29
4.2.4.16	BERT	30
4.2.4.17	Sentence-BERT	31
4.2.4.18	XLNet	32
4.2.4.19	Další	33
4.2.5	Metody učení s učitelem	33
4.2.6	Výběr metody a technologie	33
4.2.6.1	Dostupné před-trénované modely	33
4.2.7	Vyhodnocení modelů	34
4.2.8	Sestavení komponenty pro extrakci embeddingu	38
4.3	Extrakce předmětu	39
4.3.1	Návrh procesu extrakce předmětu	39
4.3.2	Extrakce textu z dokumentů	40
4.3.3	Identifikace kontextu předmětu	41
4.3.3.1	Určení počátku kontextu	41
4.3.3.2	Určení konce kontextu	44
4.3.3.3	Měření dopadu jednotlivých algoritmů a jejich parametrů	46
4.3.4	Filtrování a transformace kontextu předmětu	48
4.3.4.1	Filter prázdných řádků	50
4.3.4.2	Transformer příliš dlouhých řádků	51
4.3.4.3	Měření dopadu jednotlivých filterů a transformérů	51
4.3.5	Extrakce podle lokálních charakteristik	53
4.3.5.1	Výčet položek podle struktury odřádkování	54
4.3.5.2	Výčet položek podle odrážek	55
4.3.5.3	Meření dopadu jednotlivých extraktorů	56
4.3.6	Extrakce podle větného rozboru	56
4.3.6.1	Jazykový model	57
4.3.6.2	Nástroje pro práci s jazykem	58
4.3.6.3	Výběr technologií a implementace	59

4.3.6.4	Filtrování (ne)předmětných částí vět	60
4.3.7	Sestavení komponenty pro extrakci předmětu	63
4.4	Extrakce lokality a předmětu podnikání zadavatelů	64
5	Metrika podobnosti zakázek	65
5.1	Metrika podobnosti textu	65
5.2	Komponenty pro vyhodnocování podobnosti	66
5.2.1	Výpočet Jaccard indexu	66
5.2.2	Výpočet geografické podobnosti	67
5.2.3	Výpočet kosinové podobnosti	68
5.2.4	Výpočet „euklidovské“ podobnosti	69
6	Doporučovací systém	71
6.1	Collaborative filtering	71
6.2	Content based	72
6.3	Vyhodnocování doporučovacích systémů	72
6.4	Návrh doporučovacího algoritmu	73
6.4.1	Předmět zakázky	75
6.4.2	Lokalita zakázky	75
6.4.3	Předmět podnikání zadavatele	77
6.4.4	Kombinace vlastností	77
7	Aplikace	79
7.1	Funkce	79
7.1.1	Technologie	80
8	Budoucí rozšíření systému	81
8.1	Aktualizace dat	81
8.2	Aplikace	81
8.3	Pokročilejší algoritmy výpočtu podobnosti	82
8.4	Diverzifikace a explorace doporučování	82
8.5	Možnosti filtrování zakázek	82
8.6	Klasifikátor CPV	83
8.7	Katalog produktů	83
8.8	Vlastní model pro embedding dokumentů	83
8.9	Ladění extrakce předmětu	83
Závěr		85
Literatura		87
A Seznam použitých zkratек		93
B Obsah přiloženého CD		95

Seznam obrázků

4.1	Lokální klasifikátor pro každý uzel předka[1]	15
4.2	Lokální klasifikátor pro každý uzel[1]	15
4.3	Lokální klasifikátor pro každou úroveň[1]	16
4.4	Schéma trénování řetězového multi-label klasifikátoru[2]	17
4.5	Distibuce CPV oddílů v datasetu	18
4.6	Distibuce CPV skupin v datasetu	19
4.7	Distibuce CPV kódů v datasetu	19
4.8	Příklad reprezentace BOW[3]	22
4.9	Schéma reprezentace bag-of-n-grams[3]	22
4.10	Ukázka přechodu z BOW na LDA[3]	23
4.11	Architektury CBOW a skip-gram[4]	24
4.12	Sdílená paměť modelu paragraph vector[3]	25
4.13	Schéma skip-thought modelu	26
4.14	Schéma modelu sent2vec s příkladem[3]	27
4.15	Schéma modelu Quick-thought (b) oproti Skip-thought (a) [3]	27
4.16	Ukázka vztahu WMD (a) a WME (b) s kernelem odvozeným na základě množiny náhodných dokumentů[3]	28
4.17	Ilustrace enkodér-dekodér architektury Transformer[3]	29
4.18	Architektura paralelního učení modelu USE se sdíleným enkodérem[5]	30
4.19	Směrově orientované predikce autoregresivního jazykového modelu[6]	30
4.20	Obousměrně orientovaný autoenkodér[6]	31
4.21	Ilustrace architektury modelu BERT[7]	31
4.22	Architektura SBERT pro trénování klasifikátoru a počítání kosínové podobnosti[3]	32
4.23	Permutační modelování jazyka[6]	32
4.24	Schéma procesu extrakce předmětu dokumentace VZ	40
4.25	Příklad výpočtu ohodnocení výskytů klíčových slov	43
4.26	Histogram výskytů klíčových slov pro výběr kontextu	44
4.27	Znázornění výběru nejlepších kandidátů na kontext podle skóre	45
4.28	Ilustrace algoritmu filtru prázdných řádek	52

4.29	Ilustrace výsledku algoritmu filtru prázdných řádek	52
4.30	Ilustrace schéma UD anotace[8]	58
4.31	Ukázka CoNLL-U formátu	59
4.32	Ukázka rozkladu předmětné věty	61
4.33	Ukázka předmětné části jako vazby <i>nsubj</i>	62
4.34	Ukázka předmětné části jako vazby <i>nsubj:pass</i>	62
4.35	Ukázka předmětné části jako vazby <i>obj</i>	62
4.36	Ukázka předmětné části jako vazby <i>obl</i>	62
4.37	Ukázka předmětné části jako vazby <i>obl:arg</i>	62
4.38	Ukázka předmětné části jako vazby <i>nmod</i>	62
5.1	Ukázka průběhu geografické podobnosti 5.2	67
6.1	Schéma skládání vlastností pro cílový dotaz a referenční dataset .	74
6.2	Schéma výpočtu podobnosti	76

Seznam tabulek

3.1	Přehledová tabulka úrovní CPV kódů	10
4.1	Přehledová tabulka distribuce úrovní CPV kódů v datasetu	18
4.2	Tabulka výsledků testu <i>STSCZ</i>	37
4.3	Tabulka výsledků testu <i>STSCZ2</i>	37
4.4	Tabulka výsledků testu <i>STS16</i>	37
4.5	Tabulka výsledků testu <i>STS162</i>	38
4.6	Klíčová slova a jejich ohodnocení	41
4.7	Detekované charakteristiky a jejich dopad na multiplikativní koeficient ohodnocení klíčových slov	42
4.8	Měření dopadu jednotlivých klíčových slov na extrakci	47
4.9	Měření dopadu jednotlivých lokálních charakteristik na extrakci	47
4.10	Měření dopadu jednotlivých detektorů na extrakci	48
4.11	Měření dopadu jednotlivých filterů a transformerů na extrakci předmětných položek	53
4.12	Měření dopadu jednotlivých extraktorů (podle lokálních charakteristik) na dopad extrakce předmětných položek	56
5.1	Měření doby výpočtu algoritmů pro výpočet kosinové vzdálenosti .	68
5.2	Měření doby výpočtu algoritmů pro výpočet euklidovské vzdálenosti	69
6.1	Měření doby výpočtu algoritmů pro výpočet geografické vzdálenosti	77

Úvod

Veřejné zakázky (dále VZ) jsou nepostradatelnou součástí české státní veřejné správy. Podíl trhu veřejných zakázek na HDP České republiky dosáhl za rok 2018 11,76 % [9], což odpovídá počtu přes 61 tisíc nových zakázek v hodnotě více než 370 miliard korun[10]. Meziročně potom v České republice přibývá průměrně více než 50 tisíc nových zakázek za stovky miliard korun.

Z hlediska korupčního rizika patří veřejné zakázky k nejvíce ohroženým oblastem veřejného financování. Riziko korupce narůstá kvůli objemu transakcí a finančním zájmům zainteresovaných stran, ale také díky složitosti procesu, úzké spolupráci mezi státní a soukromou sférou a množství subjektů, které se zadávání veřejných zakázek účastní[11].

Veliký dopad na efektivitu a transparentnost řízení zakázek má elektronizace procesu zadávání, která je v dnešní době již ze zákona povinná. K tomu slouží hned několik systémů, které se navzájem doplňují nebo si konkurují.

Počet a struktura těchto systémů může být pro nezasvěceného uživatele poněkud nepřehledná. Použití stávajících systémů k vyhledávání zakázek navíc od uživatele z pravidla vyžaduje pokročilý přehled o členění a možných parametrech zakázek, což může známenat bariéru k dosažení hodnotných informací.

Ovšem i v případě, kdy má uživatel detailní přehled o problematice je stále limitován úrovní služby, kterou mu stávající systémy poskytují, respektive za kterou uživatel zaplatil. Více o omezeních stávajících systémů pojednávám v kapitole 2.1.

KAPITOLA **1**

Cíl práce

Cílem této práce je navrhnout a implementovat open-source doporučovací systém pro české veřejné zakázky. Doporučování systému bude založeno na vhodných vlastnostech veřejných zakázek, které lze z jejich dokumentace nebo portálů k tomu určených získat.

Proces extrakce vhodných vlastností je samotnou součástí práce, přičemž hlavní vlastností je zde předmět zakázky. V rámci extrakce bude provedena analýza potřebných dat zakázek, na základě které bude navržen a implementován samotný proces zpracování převážně jejich dokumentace. Extrahované vlastnosti by mely – pokud možno – reprezentovat sémantickou informaci zakázky.

Pro následnou rozšiřitelnost systému budou mít jednotlivé komponenty modulární architekturu a budou dostatečně zdokumentované pro možnost se-stavení výsledného systému.

K demonstračním účelům funkčnosti systému vznikne v rámci práce také jednoduchá webová aplikace, která bude umožňovat uživatelskou interakci se systémem a využití všech jeho důležitých vlastností, jako je vyhledávání či uživatelsky přizpůsobené zobrazování doporučovaných položek.

Všechny části systému budou navrženy a implementovány za pomoci a využití vhodných open-source technologií, přičemž samotný výsledný projekt bude distribuovaný v rámci open-source licence.

Stávající řešení

Pro uvedení do kontextu veřejných zakázek je v první řadě na místě vyjasnění co přesně veřejná zakázka je. K tomu použiji popis z Wikipedie[12], který zní:

Veřejná zakázka je nákup zboží, zadání práce, objednání díla nebo služby veřejným subjektem, kterým je stát, obec, samosprávný celek, organizace jimi založené, nebo případně dalším subjektem, který hospodaří s penězi, nebo jinými veřejnými statky nebo hodnotami pocházejícími z daní, poplatků či jiných zdrojů veřejného bohatství. Veřejné zakázky jsou realizované na základě smlouvy mezi zadavatelem a jedním či více dodavateli. Jedná se o úplatné poskytnutí dodávek či služeb nebo úplatné provedení stavebních prací. Jednou ze stran uzavírající smlouvu je veřejný zadavatel. Veřejná zakázka musí být podle zákona realizována na základě písemné smlouvy.

Ve veřejných zakázkách se tedy přirozeně nakládá s veřejnými prostředky, což na zadavatele přináší odpovědnost za její správné řízení, které je striktně vymezené zákonem. Pravidla pro řízení zakázek se v některých ohledech liší podle typu zakázky, které dělíme:

- podle předmětu na:
 1. VZ na služby,
 2. VZ na dodávky,
 3. VZ na stavební práce,
- a podle předpokládané hodnoty na:
 1. nadlimitní VZ,
 2. podlimitní VZ,
 3. VZ malého rozsahu.

2. STÁVAJÍCÍ ŘEŠENÍ

Umožnění dodržení všech pravidel řízení zakázek zajišťuje hned několik systémů, které více či méně usnadňují zadavatelům práci s jejich zadáváním.

Informační systém o veřejných zakázkách (dále jen ISVZ) poskytuje obecné informace o zadávání veřejných zakázek, informace o dodavatelích, vyhledávač zakázek či dodavatelů, základní přehled o dalších podpůrných systémech, různé datasety a statistiky.

Věstník veřejných zakázek (dále jen Věstník) je jednotným místem pro uveřejňování základních informací o veřejných zakázkách, které jsou zadávány v souladu se zákonem č. 134/2016 Sb., o veřejných zakázkách. Tyto informace jsou zadávány formou „formulářů“ (například Oznámení o zahájení zadávacího řízení, Oznámení o výsledku zadávacího řízení) skládajících se z mnoha parametrů. Věstník umožňuje parametrické vyhledávání formulářů a Profilů zadavatelů.

Profil zadavatele je podle zákona č. 134/2016 Sb., o zadávání veřejných zakázek vymezen jako elektronický nástroj, který umožňuje neomezený dálkový přístup a na kterém zadavatel uveřejňuje informace a dokumenty ke svým veřejným zakázkám. Internetová adresa nástroje je uveřejněna ve Věstníku veřejných zakázek

Kompletní seznam certifikovaných nástrojů poskytuje portál o veřejných zakázkách a koncesích¹. Jako příklad nejdůležitějších nástrojů uvádíme:

- Národní elektronický nástroj – (NEN) systém pro elektronické VZ spravovaný Ministerstvem pro místní rozvoj ČR. Z povinnosti ho užívají ústřední orgány státní správy a jejich podřízené organizace. Jeho užívání je zcela zdarma; dostupný z nen.nipez.cz
- Portál pro vhodné uveřejnění – nejpoužívanější systém pro uveřejňování a administraci VZ v ČR (využívá ho 33 % všech zadavatelů)[13], který pro vyhledávání integruje i zakázky z ostatních systémů. Aplikace nabízí různé cenové balíčky služeb obsahující možnosti správy či vyhledávání (poskytuje i balíček zdarma); dostupný z www.vhodne-uverejneni.cz
- E-ZAK – platforma umožňující nákup či pronájem konfigurovatelného nástroje, který disponuje různými možnostmi vyhledávání či poskytování informací; dostupná z www.ezak.cz
- Tender arena – nástroj umožňuje registraci profilu s možností výběru úrovně zpoplatněné služby (poskytuje zdarma registraci s omezeným použitím nástroje) pro zadávání zakázek. Vyhledávání je omezeno po jednotlivých profilech; dostupný z www.tenderarena.cz

¹Portál o veřejných zakázkách a koncesích – informační portál MMR týkající se zadávání VZ; dostupný z www.portal-vz.cz

Vedle klasických nástrojů pro zadávání VZ existují tzv. e-tržiště, která se zaměřují na zakázky malého rozsahu. Mezi nejdůležitější patří:

- TENDERMARKET – dostupný z www.tendermarket.cz,
- Gemin – dostupný z www.gemin.cz

Tenderman je vyhledávač referencí a zadávacích dokumentací[14]. Pomocí vyhledávače lze provádět jednoduchý průzkum trhu a referencí na fungující dodavatele. Vyhledávač podporuje přehledné filtrování zakázek či dodavatelů. Pro použití aplikace je nutná registrace a zakoupení licence, která se škáluje podle velikosti organizace (příležitostně nabízí časově omezenou bezplatnou registraci). Aplikace je dostupná z tenderman.cz.

VsechnyZakazky.cz je další nezávislý vyhledávač zakázek, zadavatelů a dodavatelů. Aplikace nevyžaduje, ani neumožňuje registraci. Aplikace je dostupná z www.vsechnyzakazky.cz.

Hlídač státu je nezisková organizace, jejíž cílem je transparentní státní správa. Systém funguje v podobě webové platformy, kde se na jednom místě kontrolují, analyzují, vysvětlují a propojují smlouvy z registru smluv, veřejné zakázky, dotace, detailní financování a sponzory politických stran i politiky samotné[15]. Hlídač státu disponuje silným vyhledávačem, pomocí kterého si uživatel může nastavit odběr novinek odpovídajících zadáným kritériím. Platforma je dostupná z www.hlidacstatu.cz.

2.1 Shrnutí

Přestože se některé stávající systémy snaží integrovat informace o zakázkách i z ostatních systémů, lze narazit na zakázky, které ani jejich vyhledávače nenaleznou.

Jednotlivé vyhledávače obvykle disponují různě širokými spektry parametrů pro vyhledávání zakázek. Ať už prohledávají pouze dataset vlastního systému nebo i ostatních, možnosti filtrů jednotlivých nástrojů se obvykle překrývají a jsou omezené na strukturovaná data či fulltextové vyhledávání v datech zakázek. Dle mého průzkumu žádný dostupný nástroj nedisponuje pokročilejší analýzou obsahu zakázek.

Nalezení výsledků pro uživatele je zpravidla podmíněno jeho interakcí se systémem v podobě exaktního zadání vyhledávání, což je plně dostačující v případech, kdy uživatel přesně ví, co chce nalézt. Některé systémy navíc umožňují nějakou formu odběru notifikací, které do omezené míry zastupují automatické nabízení zakázek či novinek.

2. STÁVAJÍCÍ ŘEŠENÍ

Při průzkumu stávajících systémů jsem ovšem nenašel na funkčnost, která by nabízela výsledky přizpůsobené danému uživateli ve formě automatického doporučování. Taková funkčnost by přitom byla uživateli nápomocná například v případech, kdy nemá přesnou představu o parametrech zakázky, která by ho mohla zajímat.

Pokročilejší funkčnosti současných systémů jsou navíc obvykle omezené úrovní služby, za kterou uživatel poskytovateli zaplatil. V žádném případě potom nejde o otevřené projekty ve smyslu open-source² licencování.

²Open-source – software s otevřeným zdrojovým kódem. Otevřenosť znamená jak technickou dostupnosť kódu, tak legální dostupnosť – licenci software[16].

KAPITOLA **3**

Data a doména

Myšlenka doporučovacích systémů je založena na automatickém uspořádávání potenciálně užitečných informací pro uživatele bez jeho vstupu, případně se vstupem minimálním. Každý doporučovací systém nutně potřebuje množinu dat, které – a zároveň podle kterých – bude uživateli doporučovat výsledky.

Fundamentální složkou dat doporučovacího systému v doméně veřejných zakázek jsou jejich dokumentace, které obvykle obsahují veškeré podstatné informace o zakázce. Kromě toho ale mohou být pro doporučování přínosné i další informace, které se v dokumentaci samotné vyskytovat nemusí.

Například předmět zakázky může být v dokumentaci specifikován příliš konkrétně. Tím se ztěžuje dohledání podobné zakázky, jejímž předmětem může být stejná věc jinak pojmenovaná. V tom případě by bylo užitečné takové položky předmětu kategorizovat, aby byly odkryty jejich podobnosti.

Kromě dalších vlastností by mohla být hodnotná i lokalizační podobnost či podobnost na základě předmětu podnikání zadavatele, kdy pro uživatele bude pravděpodobně zajímavější zakázka ze sousední obce (a případně v oboru uživatelova podnikání), než jí podobná zakázka z druhé strany země.

V této kapitole dále pojednávám o datech a zdrojích zmíněných vlastností využitelných v doporučovacím systému veřejných zakázek.

3.1 Dokumenty veřejných zakázek

Dokumenty veřejných zakázek jsou stěžejní datovou částí celého systému. Lze v nich dohledat většina podstatných informací potřebných pro doporučování. Zbytek potenciálně užitečných informací, které v dokumentaci nejsou zmíněné přímo, bývá nějakým způsobem často z dokumentace alespoň odkazován.

Dokumentace zakázek je zpravidla uveřejněna na profilu jejich zadavatelů, který jsou ze zákona povinni uveřejnit na certifikovaném nástroji (dále „nástroj pro profil“)[17]. Stejně tak je povinností zadavatelů zajistit, aby základní vybrané informace o veřejné zakázce měly podobu strukturovaných dat[18]. Nástroje pro profily potom tyto informace poskytují přes veřejné aplikační roz-

3. DATA A DOMÉNA

hraní (dále jen „API“), čímž se stávají data veřejných zakázek lépe dostupná pro strojové zpracování.

Základní informace o zakázkách zadavatelé uveřejňují ve Věstníku, ve kterém jsou, mimo jiné, i odkazy na profily všech zadavatelů, čímž se stává pomyslným rozcestníkem pro zpracování dat všech veřejných zakázek.

API profilových nástrojů poskytují pro každý z profilů jednotné XML rozhraní, přes které se lze dotázat na zakázky daného profilu za určité období. V poskytovaných datech jsou potom kromě základních parametrů zakázky i informace o jednotlivých uchazečích, nabízených cenách a přiložených dokumentech.

Dokumentace veřejných zakázek se běžně skládají z několika částí a v různých formátech. Často se vyskytují textové dokumenty (*doc/pdf*) jako jsou výzvy k podání nabídek, zadávací dokumentace, vzory smluv, krycí listy, oznámení o výběrech zadavatelů či finální smlouvy (obvykle podepsané a oskenované listy smluv ve formátu *pdf*). Dále se v dokumentacích vyskytují přílohy (projektové dokumentace, specifikace, dodatky...) v různých datových formátech (*doc, pdf, xls, zip*).

3.2 Kategorie produktů a služeb

Veřejné zakázky mají obecně definovaný slovník kategorií předmětu. Jsou jím tzv. CPV kódy (z angl. „Common Procurement Vocabulary“). Tento slovník je vydaný Evropským parlamentem a je tak jednotný pro všechny členské státy EU, přičemž pro každý jazyk členských států existuje mutace tohoto slovníku.

Struktura klasifikačního systému CPV se skládá z hlavního a doplňkového slovníku[19]:

1. Hlavní slovník je založen na stromové struktuře sestávající nejvýše z devítimístných kódů se slovním popisem produktu, činnosti nebo služby, které jsou předmětem smlouvy.

Úroveň	Struktura	Počet	Příklad	Popis
Oddíly	XX000000-Y	desítky (45)	45000000-7	Stavební práce
Skupiny	XXX00000-Y	nižší stovky	45200000-7	Práce pro kompletní nebo částečnou výstavbu inženýrské stavitelství
Třídy	XXXX0000-Y	vyšší stovky	45210000-7	Bytová výstavba
Kategorie	XXXXX000-Y	tisíce	45212000-7	Stavební úpravy budov sloužících pro volný čas, sporty, kulturu, ubytování a restaurace

Tabulka 3.1: Přehledová tabulka úrovní CPV kódů

Číselný kód je osmimístný a dělí se do čtyř úrovní (oddíly, skupiny, třídy a kategorie), které detailně popisují v tabulce 3.1.

Každá z posledních tří číslic kódu odpovídá dalšímu upřesnění v rámci jednotlivých kategorií. Devátá číslice slouží k ověření předešlých čísel.

Například „Stavební úpravy plováren“ mají kód 45212212-5.

Slovník takových kódů obsahuje necelých 10 tisíc.

2. K rozšíření popisu předmětu smlouvy může být použit doplňkový slovník. Položky doplňkového slovníku jsou označeny alfanumerickým kódem, kterému odpovídá slovní popis umožňující další upřesnění charakteristik nebo účelu zboží, které je předmětem nákupu. Doplňkových kódů je ve slovníku necelý jeden tisíc.

Alfanumerický kód obsahuje tři úrovně:

- a) písmeno odpovídající sekci,
- b) písmeno odpovídající skupině,
- c) tři číslice odpovídající pododdílům, přičemž poslední číslice slouží k ověření předešlých.

Například *kov* má doplňkový kód AA01 a *hliník* AA02.

Číselník těchto kódů je dostupný například na webu ISVZ[20].

Přestože zadavatelé CPV kódy zadávají při uveřejňování ve Věstníku, není zcela jednoduché k zakázkám kódy dohledat, protože samotné nástroje pro profily zpravidla tyto kódy nepodporují a Věstník neposkytuje API pro dotazování na informace o konkrétní zakázce.

CPV kódy bývají uvedené v konkrétních dokumentech VZ, ovšem zdaleka ne vždy.

3.3 Katalog produktů a služeb

Exaktně definovaná struktura kategorií CPV kódů přináší strojovému zpracování VZ různé důsledky. Přínosné je to, že se s nimi na obecné rovině jednoduše pracuje. Na druhou stranu je ovšem jejich obecnost někdy příliš omezující.

Chceme-li postoupit na další úroveň podrobnosti specifikace předmětu – na jednotlivé položky – dojdeme do situace, kdy je nutné řešit podobnost dvou položek s odlišným názvem, přestože si mohou být blízké.

K tomu by bylo zapotřebí mít vhodně strukturovaný katalog všech možných položek (produktů a služeb), podle kterého by bylo možné dohledat klasifikace jednotlivých položek na různých úrovních podrobnosti. Zde ale narazíme na problém, že tak rozsáhlý katalog, aby obsáhl všechny možné položky neexistuje. Teoreticky by mohlo jít zkombinovat katalogy z různých odvětví (jako

3. DATA A DOMÉNA

například katalogy internetových obchodů), ovšem ani takové katalogy není snadné najít, získat ani zpracovat, protože nebývají veřejně dostupné.

Jako příklad velikého katalogu uvedu internetový porovnávač cen Heureka.cz, který obsahuje více než 29 milionů produktů[21]. Heureka, jako standardní internetový obchod, neposkytuje veřejné API pro listování položek z katalogu. Jediný možný způsob jak standardně přistoupit k jejich obsahu je tedy přes webové rozhraní, které ovšem nepodporuje rozsáhlejší stránkování než 20 položek na stránku. To znamená, že samotné výlistování všech položek pro načtení celého katalogu s názvy produktů by znamenalo nejméně 1,5 milionu požadavků a za stabilní odezvy 0,3 vteřiny na požadavek by se tak při sériovém stahování samotný katalog stahoval déle než 5 dní. Navíc za předpokladu, že jedna položka bude mít název dlouhý zhruba 100 znaků, budou data skládající se z pouhých názvů položek dosahovat řádu jednotek gigabajtů.

3.4 Data o lokacích

V úvodu této kapitoly zmiňuji také místní zařazení zakázek, které by doporučování mohlo pozitivně obohatit o upřednostňování „blízkých“ zakázek oproti „dalekým“.

Každý zadavatel musí uvádět adresu sídla své organizace, přičemž takové informace o ekonomických subjektech jsou poté dostupné v různých rejstřících.

Jedním ze systémů poskytujících informace o všech ekonomických subjektech je ARES³, který kromě webového vyhledávače poskytuje také API pro dotazování na jednotlivé subjekty podle čísla IČO. Tím lze získat lokalizační informaci o zadavatelích VZ.

Počítání podobnosti (respektive inverzně vzdálenosti) adres samotných nedává kloudný smysl. Pro doporučování je tak nutné převést adresy na souřadnicový systém (GPS), ve kterém jsou takové operace přirozené. Proces převedení adresy na zeměpisné souřadnice se nazývá „geokódování“[23] a poskytuje ho jako službu většina mapových platform.

Například platforma Mapy.cz poskytuje API pro jednoduché dotazování na GPS souřadnice konkrétní adresy.

3.5 Data o předmětu podnikání zadavatelů

Posledním zmíněným typem informace je předmět podnikání zadavatele.

Tyto informace též poskytuje systém ARES a lze tak k nim přistoupit stejným způsobem jako k adresám.

³ARES - Administrativní registr ekonomických subjektů je informační systém, který umožňuje vyhledávání nad ekonomickými subjekty registrovanými v České republice. Zprostředkovává zobrazení údajů vedených v jednotlivých registrech státní správy, ze kterých čerpá data[22].

Extrakce vlastností zakázek

Ne všechna data veřejných zakázek (viz. kapitola 3.1) jsou z hlediska strojového zpracování strukturovaná a tedy lehce zpracovatelná. Například dokumentace zakázek obsahují soubory v mnoha různých datových formátech, různé kvalitě a jejich obsah je převážně v podobně souvislého nestrukturovaného textu.

Aby byl doporučovací algoritmus sytému schopný s daty zakázek efektivně pracovat, je nutné provést jejich předzpracování ve smyslu extrakce důležitých vlastností jak z jejich strukturované, tak i nestrukturované části.

O jednotlivých částech extrakce vlastností zakázek pojednávám dále v podkapitolách této kapitoly.

4.1 Klasifikace dokumentů

V případě, že původní data používané doporučovacím systémem nedisponují kompletně všemi vlastnostmi, které jsou k doporučování přínosné či nutné, je na místě jejich automatické doplnění (klasifikování či regrese).

U veřejných zakázek jsou jednou takovou potenciálně vhodnou vlastností dříve zmiňované CPV kódy.

Klasifikace CPV kódů se ovšem ukazuje být hned ze tří důvodů problematická.

Zaprvé je to jejich hierarchická struktura. Standardní klasifikátory založené na strojovém učení jsou zpravidla schopné klasifikovat pouze ploché struktury tříd. Pro klasifikování hierarchické struktury tříd je tak nutné aplikovat nadstandardní postupy. O možnostech řešení klasifikování hierarchických struktur pojednávám více v samostatné kapitole 4.1.1.

Zadruhé, zakázky mohou zasahovat hned do několika odvětví a jejich zařazení tak není omezené na jediný CPV kód. V případech, kdy klasifikované položky mohou spadat do více než jedné třídy mluvíme o tzv. „multi-label“ klasifikaci, o které se dále rozepisují v kapitole 4.1.2.

4. EXTRAKCE VLASTNOSTÍ ZAKÁZEK

Zatřetí je problematické množství všech kódů (tříd pro klasifikaci). Intuitivně, čím více je tříd ke klasifikaci, tím více je potřeba vzorků pro trénování klasifikátoru. CPV kódů jsou tisíce, tudíž pro korektní klasifikaci všech kódů je potřeba veliká množina dat s objektivním rozložením tříd. Podrobnější analýzu distribuce CPV kódů v datech řeším v kapitole 4.1.3.

4.1.1 Klasifikace hierarchických struktur

Zatímco lidská mysl vnímá přirozeně okolní svět v hierarchických strukturách, architektury modelů strojového učení jsou zpravidla navržené pro plochou reprezentaci dat. Proto je v praktických úlohách často nutné řešit překlenutí nesouladu těchto dvou pohledů. Podle článku[1] se nabízí několik základních přístupů, jak tento problém řešit.

Autorka článku ilustruje jednotlivé přístupy na příkladě druhů domácích mazlíčků, jako jsou vidět obrázku 4.1.

1. Plochá klasifikace

Přímočaré řešení spočívající v opomenutí hierarchické struktury a klasifikaci pouze listových tříd. Není zapotřebí řešit nic navíc, stačí přímo použít standardní plochý klasifikátor. Toto zjednodušení je ovšem za cenu ztráty potenciálně důležitých informací v podobě vazeb samotné struktury.

2. Globální klasifikace

Globální klasifikace spočívá v navržení jednoho komplexního–globálního modelu pro podchycení celé hierarchie jako celku. Jednotlivé implementace se mohou velice lišit. Globální model může být založen na tzv. „clusterování“, multi-label klasifikaci nebo různě zkombinovaných algoritmů uzpůsobených na konkrétní případ užití.

3. Hierarchicky strukturovaná lokální klasifikace

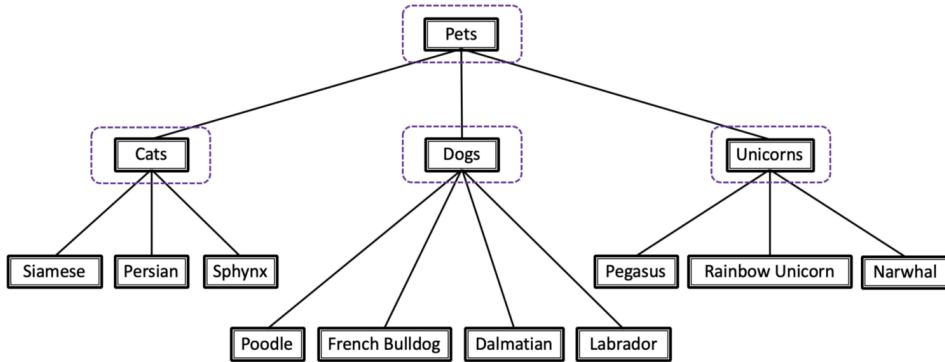
Tento přístup se zakládá na myšlence použití několika oddělených klasifikátorů pro různé pozice ve struktuře. Existují tři možnosti, jak tento přístup implementovat:

- Lokální klasifikátor pro každý uzel předka

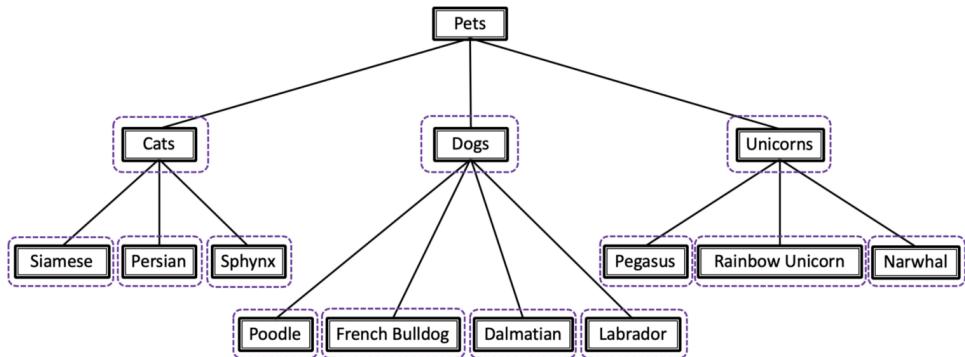
Pro každý z vnitřních uzlů se natrénuje multi-label klasifikátor pro odlišení tříd jeho potomků. Tento přístup jde vyladit tak, aby se klasifikovaly jen relevantní třídy pro daný uzel a nedocházelo k chybným klasifikacím sourozeneckých tříd. Znázornění přístupu je zobrazeno na obrázku 4.1.

- Lokální klasifikátor pro každý uzel

Přístup se zakládá na vytvoření binárního klasifikátoru pro každý uzel struktury (kromě kořenu). Oproti předchozímu přístupu lokální



Obrázek 4.1: Lokální klasifikátor pro každý uzel předka[1]



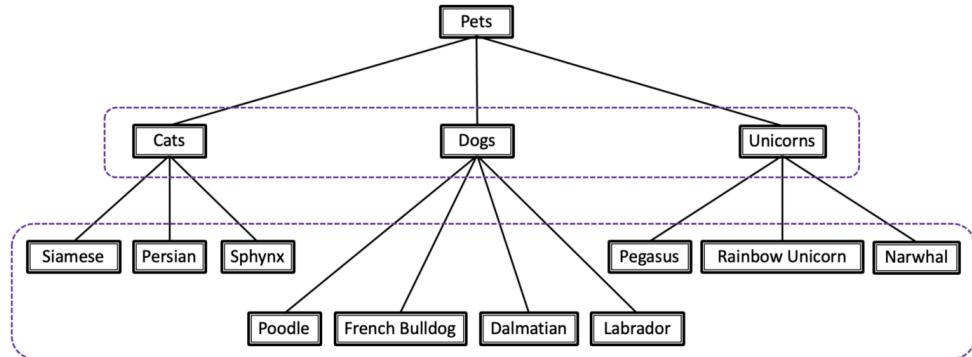
Obrázek 4.2: Lokální klasifikátor pro každý uzel[1]

klasifikátory odpovídají na otázku „Odpovídá tato třída instanci?“ místo otázky „Která z následujících tříd instanci odpovídá?“ Tato metoda je zachycena na obrázku 4.2.

- Lokální klasifikátor pro každou úroveň

U tohoto přístupu se množství klasifikátorů omezuje na počet úrovní hierarchické struktury. Z principu zde může docházet k nesmyslným klasifikacím sourozeneckých tříd. Přístup je zobrazen na obrázku 4.3.

Přístup pomocí lokálních klasifikátorů je intuitivní a ponechává hierarchickou informaci za použití standardních klasifikátorů. Může se ovšem projevit problém se zpětnou propagací chyb, která v základní implementaci mezi jednotlivými klasifikátory neprobíhá.



Obrázek 4.3: Lokální klasifikátor pro každou úroveň[1]

4.1.2 Multi-label klasifikace

Multi-label klasifikace je úloha „přiřazování“ jednotlivých tříd (příznaků) dané instanci. Třídy jedné instance tak nejsou vzájemně výlučné, na rozdíl od tzv. „multi-class“ klasifikace, ve které jde o úlohu „rozřazování“ jednotlivých instancí do tříd.

Existuje několik přístupů řešení multi-label klasifikace[2]:

1. Jeden z několika

Přístup „jeden z několika“ v podstatě převádí problém multi-label klasifikace na standardní multi-class tím, že pro každou třídu se vytvoří jeden klasifikátor a ve výsledku se vybírá třída toho klasifikátoru, který dosahuje největší jistoty.

2. Binární přítomnost

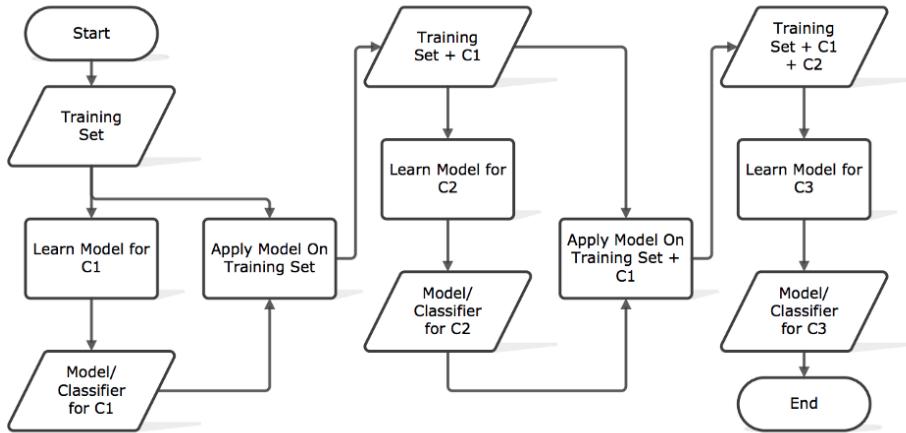
V tomto případě se vytváří binární klasifikátor pro každou třídu, přičemž ve výsledku se vybírají všechny třídy pozitivně klasifikované na svou přítomnost. Instanci tak nemusí být přiřazena ve výsledku žádná třída. Tato metoda nebude ohled na vzájemné vazby mezi třídami.

3. Řetězec klasifikátorů

Přístup spočívá ve vytvoření řetězce klasifikátorů, kde každý jednotlivý klasifikátor využívá výsledků jemu předcházejících klasifikátorů. Tato metoda je schopná podchytit vazby mezi třídami. Počet jednotlivých klasifikátorů je opět rovný počtu tříd, ale trénování je sofistikovanější, jako je zobrazeno na obrázku 4.4.

4. Podmnožiny tříd

Přístup zakladající se na podmnožinách celého setu tříd je také schopný podchycovat vazby mezi třídami. Vytvoří se klasifikátor pro každou



Obrázek 4.4: Schéma trénování řetězového multi-label klasifikátoru[2]

z umělých tříd odvozených jako podmnožiny původních tříd. U této metody dochází k problému tzv. „kombinatorické exploze“, kdy úlohu nelze technologicky dořešit.

5. Adaptivní algoritmy

Metoda adaptivních algoritmů spočívá v uzpůsobování „single-label“ klasifikačních algoritmů na multi-label změnou jejich základních funkcí. Více ve zdrojovém článku[2].

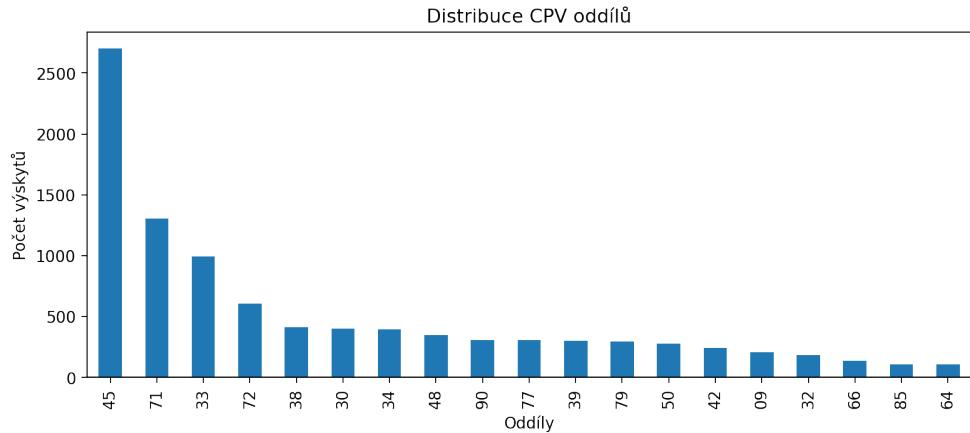
4.1.3 Analýza CPV kódů v datech

Jedním ze základních kroků data science projektů předcházejících samotnému budování modelu je analýza dat. Tuto kapitolu tak věnuji statistické analýze CPV kódů v datech VZ.

Dataset s přiřazenými CPV kódy k zakázkám mi byl poskytnut správci portálu Hlídač státu. Obsahuje přes 30 tisíc záznamů páru evidenčních čísel zakázek(dále jen ECZ⁴) a jim přiřazené CPV kódy. Zakázky datasetu jsou náhodně vybrané zakázky s uvedeným ECZ(zhruba pětina ze všech zakázek) za období leden až červen 2019 od více než tisíce různých zadavatelů z celé České republiky. Majoritní zastoupení (přes 12 tisíc zakázek) má samotný zadavatel Jihočeského kraje a druhý nejpočetnější je zadavatel Libereckého kraje s jedním tisícem zakázek. Nad 500 zakázek v datasetu dosahují ještě další 4 zadavatelé.

⁴Evidenční čísla zakázky jsou identifikátory zakázek používané ve Věstníku. Napříč profily zadavatelů ovšem zpravidla používané nejsou.

4. EXTRAKCE VLASTNOSTÍ ZAKÁZEK



Obrázek 4.5: Distribuce CPV oddílů v datasetu

Některé ECZ se v datasetu vyskytují mnohonásobně, zatímco jiné nemají přiřazený žádný CPV kód a některé mají kódy nevalidní. Dataset tak čistím pro ponechání pouze validních záznamů. Ve výsledku tak zbývá 10 607 607namů párů kódů ECZ–CPV obsahujících 9127 unikátních ECZ kódů (počet zakázek) a 1756 unikátních kódů CPV.

V tabulce 4.1 zaznamenávám hodnoty distribuce CPV kódů v datasetu. Na diagramech odkázaných z tabulky jsou zobrazeny odpovídající distribuce relevantních tříd:

Úroveň	Počet unikátních	Počet relevantních*	Diagram
Oddíly	44	19	4.5
Skupiny	245	21	4.6
Třídy	636	17	—
Kategorie	1134	9	—
Celý kód	1756	8	4.7

Tabulka 4.1: Přehledová tabulka distribuce úrovní CPV kódů v datasetu;

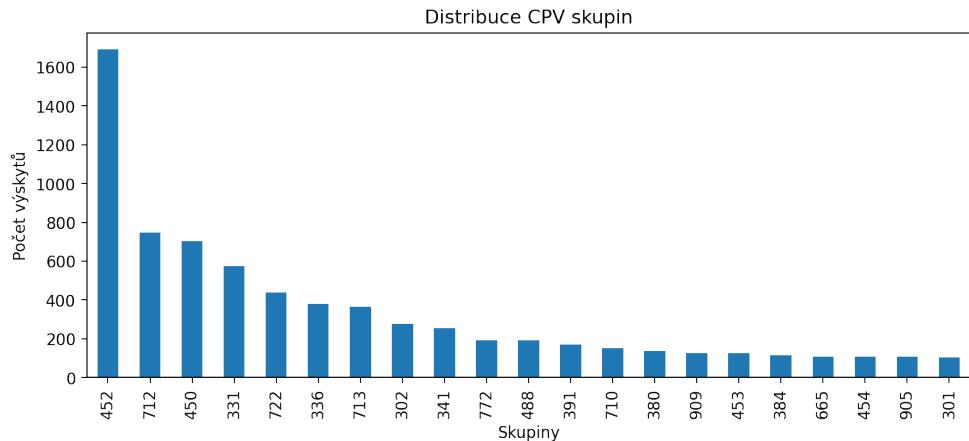
* za relevantní třídy beru takové, které mají alespoň 100 výskytů

Z hodnot v tabulce 4.1 a diagramů 4.5, 4.6 a 4.7 je zřejmé, že zastoupení tříd je nevyrovnané. Zatímco zakázky týkající se stavebního odvětví (oddíl 45) tvoří signifikantní majoritu, některé oddíly nemají zastoupení vůbec (oddíl 41 – Shromážděná a upravená voda). Relevantních⁵ tříd je necelá polovina.

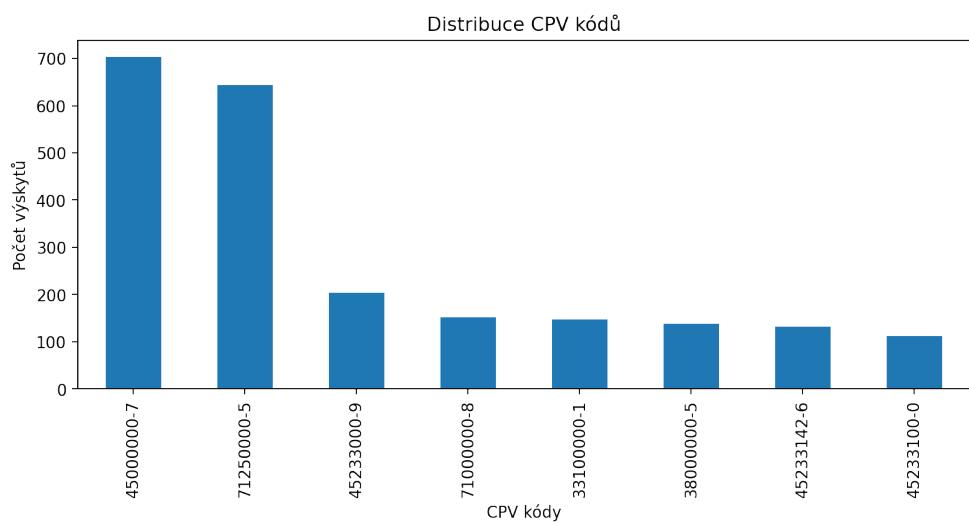
Vzhledem k celkovému počtu CPV kódů (necelých 10 tisíc) oproti počtu unikátních kódů v datasetu (1756) je zřejmé, že konkrétních kódů chybí veliké množství. Pouhých 8 z nich potom čítá více než 100 výskytů.

⁵Relevantní – třídy se zastoupením alespoň 100 výskytů

4.1. Klasifikace dokumentů



Obrázek 4.6: Distribuce CPV skupin v datasetu



Obrázek 4.7: Distribuce CPV kódů v datasetu

4. EXTRAKCE VLASTNOSTÍ ZAKÁZEK

Je možné si povšimnout, že některé oddíly samy o sobě obsahují hned několik relevantních skupin, což vysvětluje vyšší počet relevantních skupin oproti počtu relevantních oddílů. Tento jev je potom vidět na diagramu 4.6, kde je hned několik skupin z oddílů 45, 71, 33 a 30.

Diagram 4.7 potom ukazuje, že mnoho zakázek má uvedené CPV zařazení v nejmenším možném detailu, což potvrzuje počet výskytů kódů 45000000-7, 71000000-8 a 38000000-5.

Na základě zjištění analýzy datasetu usuzuji, že trénovat úspěšně klasifikátor pro určení CPV zařazení zakázek podle textu jejich dokumentace by na množině zakázek tohoto datasetu nebylo možné. Pro smysluplné učení klasifikátoru by dataset musel být mnohem obsáhlnejší a vyvážený v zastoupení jednotlivých tříd.

4.1.4 Extrakce CPV kódů

Přestože v systému neaplikuji automatickou klasifikaci dokumentů, některé CPV kódy lze získat přímo z dokumentů.

Pro tyto účely implementuji extraktor CPV kódů z textu, který podle regulárního výrazu vyhledává všechny řetězce odpovídající formátu CPV kódu.

4.2 Embedding dokumentů

Dnešní digitální výpočetní technologie spoléhají bez výjimky na číselnou reprezentaci dat. Některá data je možné přirozeným způsobem zobrazovat do spojitého prostoru, jako například obraz reprezentovaný dvourozměrným popisem barev, či zvuk popsaný průběhem signálu, zatímco jiná data takto přirozeně reprezentovat nelze.

Jedním z obtížně reprezentovatelných typů dat je text přirozeného jazyka, kde se pro reprezentaci slov při jeho zpracování tradičně užívá kód 1 z n, který z principu vytváří velmi řídce obsazené prostory s vysokým počtem dimenzí. To je z mnoha důvodů nepraktické. Data jsou prostorově náročná a reprezentace nijak nezachycuje vztahy mezi slovy.

Žádoucí je tak převést všechna slova do spojitého prostoru pevně dané dimenze, která nebude závislá na velikosti slovníku.

V praxi se používají vektory v prostoru reálných čísel nabývajícím desítky až stovky rozměrů. Vnoření jsou vytvářena automaticky, často s použitím strojového učení, čímž se význam jednotlivých rozměrů stává neinterpretovatelný. Vektory tak mají význam pouze ve vztahu k ostatním, samostatně ne.

4.2.1 Embedding

Ve svém článku Palachy[3] shrnuje vnořování slov (dále angl. embedding), jako metodu reprezentace slov v číselném vektorovém prostoru, která se stala

nedílnou součástí dnešních řešení úloh strojového zpracování přirozeného jazyka (dále NLP z angl. natural language processing). Embedding umožňuje modelům strojového učení závisejícím na vektorové reprezentaci vstupu využít takové reprezentace jako bohatšího vyjádření samotného vstupního textu. Ve vektorovém prostoru je možné zachovat více sémantické i syntaktické informace, což napomáhá k dosažení lepších výsledků v téměř kterékoli úloze zpracování přirozeného jazyka.

Milníkem se stala v roce 2013 publikace práce skupiny Tomáše Mikolova. Jejich word2vec model s inovativním přístupem k získávání embeddingů spustil vlnu zájmu o obor. Palachy dodává, že myšlenka embeddingu a její zásadně pozitivní dopady vedly vědce k úvaze o její aplikaci na větší textové celky, od vět až po knihy. Snaha mnoha lidí vyústila v řadu nových metod získávání vektorové reprezentace s různými inovativními řešeními a další významné průlomy v oboru. Ve článku [3] člení přístupy řešení text embeddingu do čtyř kategorií.

1. Sumarizace slovních vektorů

Klasický přístup, který zastupuje například algoritmus „bag-of-words“ v případě „one-hot“ slovních vektorů. Je možné aplikovat různá schémata vážení k summarizaci vektorů.

2. Modelování témat

Přestože zde nelze mluvit o získávání embeddingů jako o hlavním účelu, modelovací techniky jako LDA („latent Dirichlet allocation“) nebo PLSI („probabilistic latent semantic indexing“) inherentně generují prostor pro embedding dokumentů. Tím modelují a vysvětlují distribuci slov v korpusu, kde jednotlivé dimenze mohou být viděny jako latentní sémantické struktury skryté v datech.

3. Enkodér—dekodér modely

Přístup využívající vnitřní reprezentaci dat v podobě číselných vektorů. Modely vznikají učením bez učitele s výhodou použití stále dostupnějších velkých korpusů s neoznačenými daty.

4. Učení s učitelem

Modely neuronových sítí mají schopnost naučení získávat obohacené reprezentace vstupních dat, které využívají k řešení úloh souvisejících s textem. Takto naučené sítě obsahují skryté vrstvy, kde jsou data reprezentovaná právě číselnými vektory.

4.2.2 Klasické metody

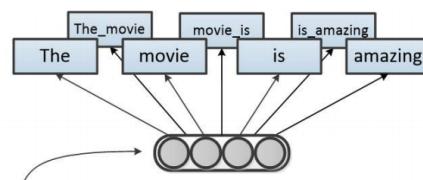
4.2.2.1 Bag-of-words

Metoda reprezentující text jako (multi)množiny vyskytujících se slov – tzv. „bag-of-words“ (dále BOW). Každý dokument je zastoupen vektorem o délce velikosti slovníku, kde každá pozice zastupuje počet výskytů daného slova. Z této podstaty metoda neuchovává žádnou gramatickou ani souslednou informaci o textu. Viz. obrázek 4.8.

the dog is on the table

0	0	1	1	0	1	1	1
are	cat	dog	is	now	on	table	the

Obrázek 4.8: Příklad reprezentace BOW[3]



Obrázek 4.9: Schéma reprezentace bag-of-n-grams[3]

4.2.2.2 Bag-of-n-grams

Metoda se snaží oproti klasickému BOW omezit ztrátu informace o pořadí. Místo výskytu slov tak počítá výskyty n-slových sousloví. Klasický BOW je tak případ této metody pro $n = 1$. Hlavní problém této metody je nelineární závislost velikosti slovníku na počtu unikátních slov. Proto se často užívají techniky ke snížení velikosti slovníku. Viz. obrázek 4.9.

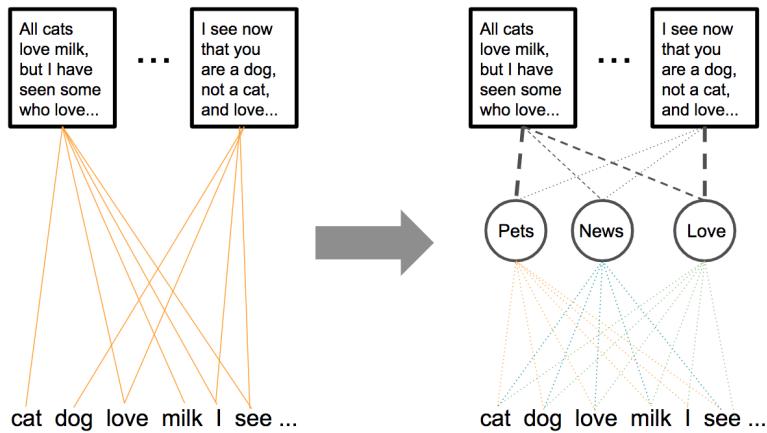
4.2.2.3 tf-idf

Předešlé metody počítají jednotlivé výskyty, což není zcela objektivní metrika, protože některá slova se vyskytují obecně častěji, čímž se stávají méně důležitými. Alternativou je vážící schéma „term frequency – inverse document frequency“ (dále *tf-idf*). Toto schéma se skládá ze dvou složek: četnost slova v dokumentu (*tf*) a převrácená četnost slova ve všech dokumentech (*idf*). Pro výsledné ohodnocení se složky násobí, přičemž ve výpočtu *tf* roste s počtem výskytů, zatímco *idf* je vysoké, když je slovo vzácné.

4.2.3 Modelování témat

4.2.3.1 Latent Dirichlet allocation

LDA je generativní statistický model opírající se o myšlenku, že „každý dokument lze popsat distribucí témat a každé téma může být popsáno distribucí slov“. Při vytváření modelu z korpusu dokumentů vzniká skrytá (latentní) vrstva abstraktních témat. Každý dokument je poté reprezentován jako výběr z Dirichletova rozdělení nad tématy a každé téma jako výběr z Dirichletova rozdělení nad slovy. Hlavním užitím LDA je neřízené odhalování témat v do-



Obrázek 4.10: Ukázka přechodu z BOW na LDA[3]

kumentech, tzv. modelování témat (z angl. topic modelling). Existují avšak i užití, kde latentní prostor témat se využívá jako prostor pro embedding dokumentů. Viz. obrázek 4.10.

4.2.4 Metody učení bez učitele

Většina modelů text embeddingu byla navržena na základě myšlenky distribuční hypotézy (z angl. The Distributional Hypothesis), podle které slova vyskytující se ve stejných kontextech tíhnou k podobnému významu. Tuto myšlenku o slovech modely dále rozšiřují různými přístupy i pro delší části textu.

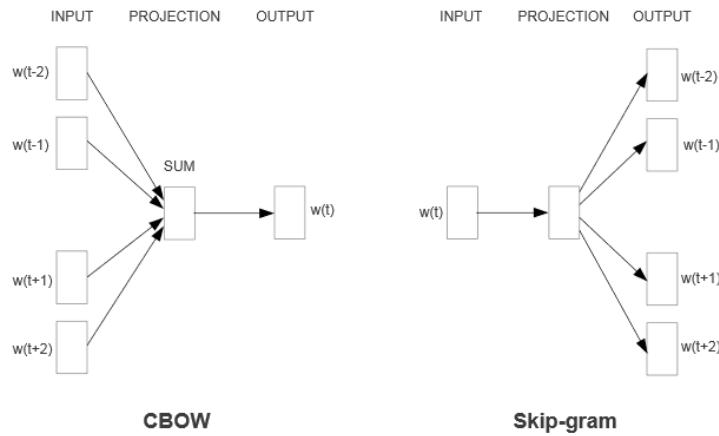
4.2.4.1 word2vec

Jak už bylo řečeno úvodní kapitole, word2vec[4] zaznamenal převrat v oboru zpracování přirozeného jazyka. Word2vec je mělká (dvouvrstvá) neuronová síť, která dokáže matematicky podchytit vazby mezi slovy z korpusu. Jako příklad úspěšného objevu se uvádí široce známá formula:

$$word2vec('king') - word2vec('man') + word2vec('woman') \cong word2vec('queen') \quad (4.1)$$

Tvůrci modelu přichází se dvěma architekturami. První je CBOW (z angl. continuous bag-of-words), která predikuje prostřední slovo na základě několika okolních slov, zatímco druhá, tzv. skip-gram architektura, v jistém smyslu dělá opačný proces, tedy predikuje okolní kontextová slova na základě jednoho vstupního. Dle autorů je mezi architekturami výkonnostní rozdíl. Zatímco CBOW je rychlejší, architektura skip-gram dokáže predikovat lépe pro méně obvyklá slova[4]. Na obrázku 4.11 jsou vidět obě architektury.

4. EXTRAKCE VLASTNOSTÍ ZAKÁZEK



Obrázek 4.11: Architektury CBOW a skip-gram[4]

4.2.4.2 n-gram embeddingy

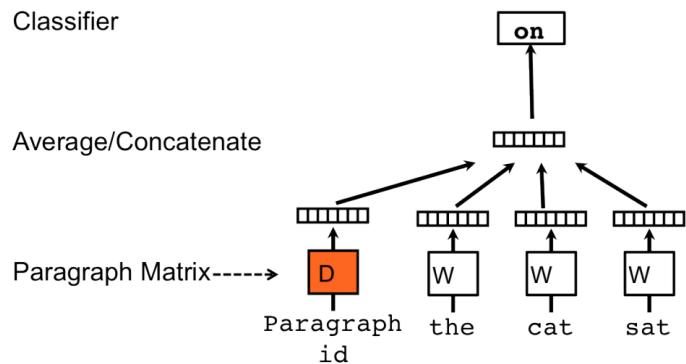
V práci [24] autoři rozšiřují skip-gram algoritmus modelu word2vec pro použití na krátkých frázích. Místo jednoslovných tokenů tak při učení modelu používají několikaslovné fráze. Takový přístup je přirozeně nevhodný pro delší fráze kvůli rychlému růstu velikosti slovníku se zvyšující se délkou fráze. Metoda také negeneralizuje svou funkci na neznámé fráze jako jiné metody.

4.2.4.3 Agregace embeddingů slov

Velice intuitivní metoda pro získání embeddingu delšího textu je agregace embeddingů jednotlivých slov. Provedením vektorové operace získáme opět vektor v tom samém embeddingovém prostoru. Nabízí se například sčítání či průměrování vektorů, ale existují i o něco složitější řešení obsahující další kroky či vrstvy. Například v práci Kentera[25] návrhli jednoduchou neuronovou síť nad průměrovánými slovními embeddingy, čímž predikuje okolní věty. Palachy uvádí ve svém článku[3] další příklady.

4.2.4.4 doc2vec

Model, zvaný jako „paragraph vectors“[26], je zřejmě první pokus zobecnění použití modelu word2vec na sekvence slov. Metoda je založená na rozšíření standardního modelu o paměťový vektor, který cílí na zachycení téma/kontextu ze vstupu. Každý odstavec je mapovaný na unikátní vektor (tzv. paragraph vector) podobně jako slova ze slovníku. Tento vektor je sdílený mezi všemi kontexty okna plovoucího přes odstavec. Pro predikované slovo se paměťový



Obrázek 4.12: Sdílená paměť modelu paragraph vector[3]

vektor přidává ke kontextu fixní velikosti. Vektory pro neznámé odstavce jsou náhodně inicializované. Paragraph vector je zobrazen na schéma 4.12.

4.2.4.5 GloVe

V práci Penningtona[27] přichází s odlišnou myšlenkou získávání informace pro embedding slov, kterou předznamenává jeho název odvozený z anglického „Global Vectors“. Zatímco word2vec model je zaměřen pouze na lokální informaci v textu (slova z okolí), GloVe vedle lokální zachycuje i globální statistiky korpusu. Ve výpočtech k tomu zahrnuje matici společných výskytů slov (z angl. co-occurrence matrix). Autorům se tak podařilo vytvořit další model, který předčil ostatní modely v úlohách rozpoznávání pojmenovaných entit, podobnosti či analogie slov.

Ve článku Ganegedara[28] je GloVe popsán dopodrobna.

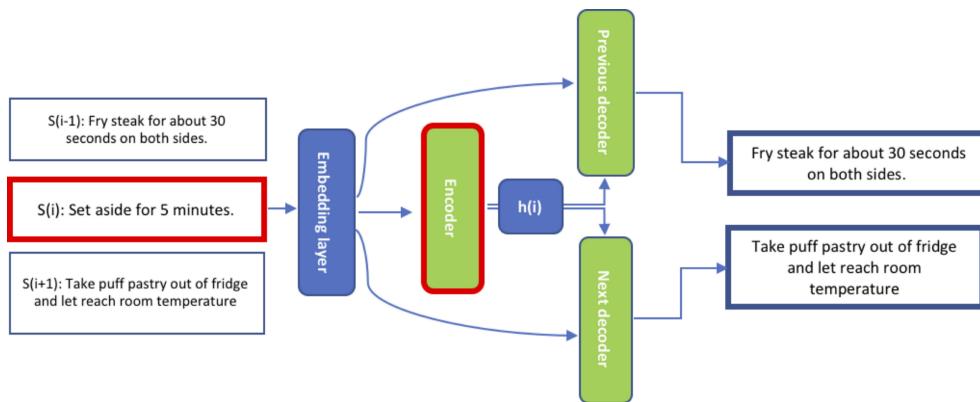
4.2.4.6 Skip-thought vektory

Další pokus o zobecnění skip-gram architektury word2vec modelu rozšiřuje původní koncept ve smyslu použití celé věty jako základní jednotky pro predikci. Skip-thought přístup predikuje předcházející a následující větu. Autoři používají rekurentní neuronové sítě pro enkodér a dekodér, které využívají embedovací vrstvu provádějící embedding jednotlivých slov. Více je vysvětleno na obrázku 4.13.

4.2.4.7 FastSent

Hill ve své práci[29] navrhl zjednodušené řešení skip-thought modelu, kde místo enkodéru a dvou dekodérů se používá klasický BOW. Díky tomu je dosaženo značného snížení výpočetní náročnosti.

4. EXTRAKCE VLASTNOSTÍ ZAKÁZEK



Obrázek 4.13: Schéma skip-thought modelu – věta $s(i)$ je zakódována enkodérem do skryté reprezentace $h(i)$, na základě které dekodéry predikují předešlou $s(i-1)$ a následující $s(i+1)$ větu[3]

4.2.4.8 sent2vec

Představený model sent2vec v Pagliardiniho[30] práci dále kombinuje předešlé přístupy. Autoři rozšiřují CBOW algoritmus modelu word2vec o schopnost zpracování celých vět. Pro predikci slova ve větě je místo kontextového okna o fixní velikosti použito jako kontext celý zbytek věty. Vektory slov z kontextu jsou opět průměrované jako vstup do neuronové sítě. Schéma modelu je zobrazeno na obrázku 4.14.

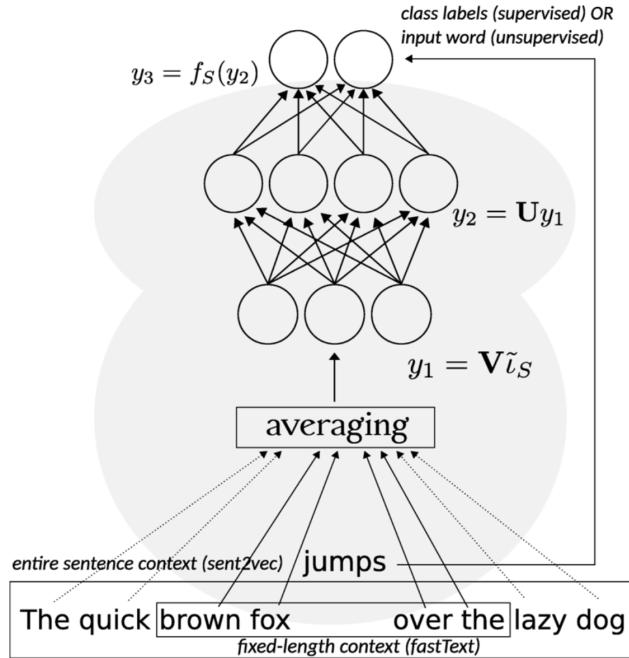
4.2.4.9 Quick-thought vektory

Skupina pod vedením Logenswarana[31] navrhla přeformulováním úlohy embeddingu dokumentů zcela nový přístup. Kontext, ve kterém se věta vyskytuje, začali predikovat metodou učením s učitelem. Viz. obrázek 4.15.

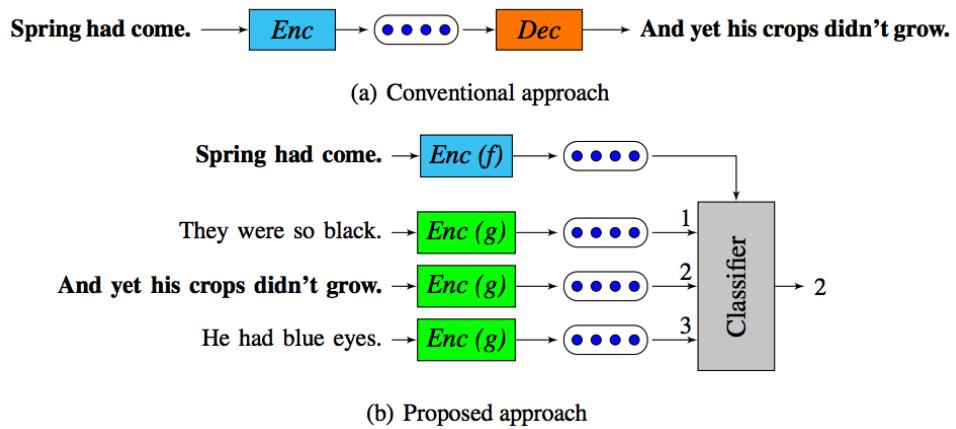
4.2.4.10 Word Mover's Embedding

Metoda je založená na „Word Mover's Distance“[32] – metrice pro podobnost mezi dvěma dokumenty definováná jako minimální vzdálenost, kterou potřebují embedovaná slova jednoho dokumentu urazit ve vektorovém prostoru k dosažení embedovaných slov druhého dokumentu. K těmto účelům bylo navrženo D2KE (z angl. distances to kernel and embeddings)[33], jako metodologii pro odvození kernelu z dané vzdálenostní funkce. Na obrázku 4.16 je zobrazeno odvození takového kernelu.

Více do detailu Palachy popisuje ve článku[3].

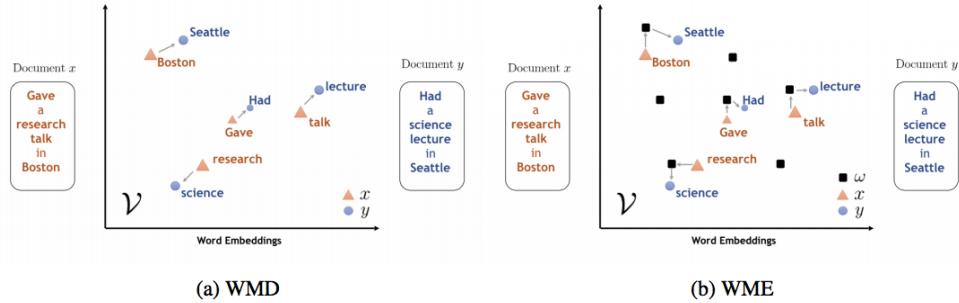


Obrázek 4.14: Schéma modelu sent2vec s příkladem[3]



Obrázek 4.15: Schéma modelu Quick-thought (b) oproti Skip-thought (a) [3]

4. EXTRAKCE VLASTNOSTÍ ZAKÁZEK



Obrázek 4.16: Ukázka vztahu WMD (a) a WME (b) s kernelem odvozeným na základě množiny náhodných dokumentů[3]

4.2.4.11 ELMo

V práci Peters a spol.[34] přichází s myšlenkou hluboce „kontextualizovaných“ slovních embeddingů. Místo přiřazování fixních embeddingů každému slovu, ELMo (z anglicky Embeddings from Language Model) používá oboustrannou LSTM (z anglicky bi-directional long short-term memory) síť ke zpracování celé věty, čímž vytváří embedding pro slova na základě kontextu z obou stran. Vnitřní reprezentace slov je založená na jednotlivých znacích, čímž se model stává robustnějším pro slova mimo slovník použitý při trénování.

Jay Alammar ve svém článku[7] přehledně prezentuje celý proces ELMo embeddingu.

4.2.4.12 ULMFiT

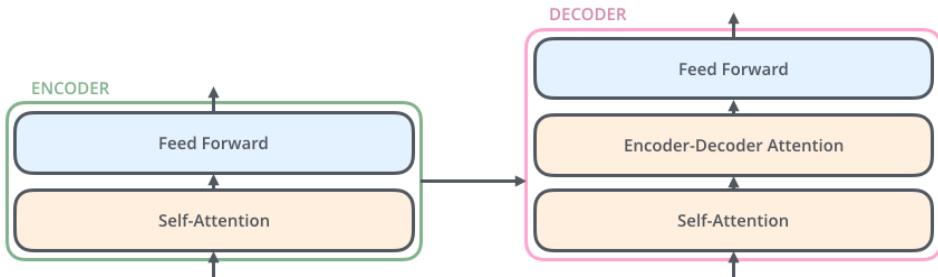
Zde se nejedná přímo o metodu embeddingu, jako spíš o inovativní přístup k řešení úloh zpracování přirozeného jazyka. ULMFiT (z anglicky Universal Language Model Fine-tuning)[35] přichází jako první s aplikací metody tzv. „transfer-learningu“ v NLP. Metoda spočívá ve třech krocích:

1. Základní před-trénování jazykového modelu (například na textu z Wikipedie).
2. Ladění (z anglicky fine-tune) jazykového modelu na cílové doméně.
3. Ladění klasifikátoru pro cílovou úlohu.

Podle článku[36], tímto způsobem použitá síť výrazně překonala dosavadní state-of-the-art řešení na několika úlohách klasifikace textu.

4.2.4.13 Transformers

Práce „Attention Is All You Need“[37] odstartovala novou éru v oboru zpracování přirozeného jazyka. Vznikla rodina modelů typu tzv. Transformer,



Obrázek 4.17: Ilustrace enkodér-dekodér architektury Transformer[3]

který se vyhýbá konvolučnímu i rekurentnímu přístupu ke zpracování sekvence a místo toho používá inovativní přístup zvaný „attention“. Stále však uchovávají architekturu enkodér (čtení vstupu) – dekodér (provádění predikce).

Tyto modely generují kontextové embeddingy obsahující informaci o sousedních slovech, přestože jejich cílem není tvorit bohatý embedovací prostor pro vstupní text. Na obrázku 4.17 je zobrazeno schéma enkodér–dekodér architektury Transformer.

4.2.4.14 USE

„Universal Sentence Encoder“[38] přichází s unikátní myšlenkou učení enkodéru pro získání embeddingů vět. USE je založený na skip-thought modelu, avšak místo obou částí enkodér–dekodér architektury je použito pouze architektury se sdíleným enkodérem, který je paralelně učený na různých úlohách. Tím autoři cílí na vytvoření jednoho univerzálního enkodéru, schopného plnit roli embedování vět v různých aplikacích jako je klasifikace, klastrování či podobnost textu.

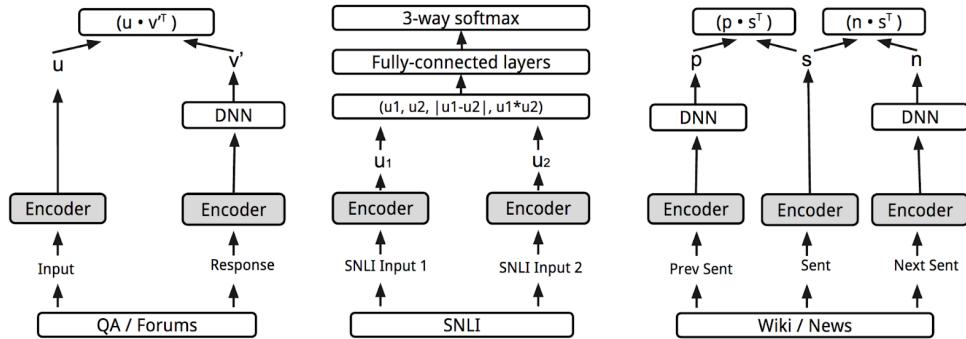
Autoři USE přichází se dvěma možnostmi řešení enkodéru. První používá tzv. hluboce průměrovanou síť (z angl. deep averaging network), zatímco druhá se opírá o složitější strukturu transformeru[5]. Na obrázku 4.18 je zobrazena architektura sdílení enkodéru.

4.2.4.15 GPT

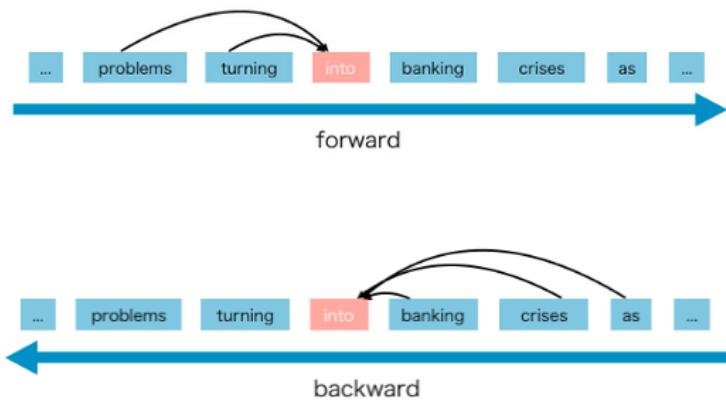
Jako jednou z prvních adaptací Transformer architektury je GPT (z angl. Generative Pre-Training Transformer)[39]. Oproti základní architektuře používá však jinou strukturu. GPT se skládá pouze z na sebe napojených bloků dekodéru. Vyškálováním modelu autoři později dosáhli lepšího modelu, pojmenovaného jako GPT-2[40].

Ve svém článku[6] Liang uvádí, že GPT se řadí mezi tzv. autoregresivní jazykové modely, které používají kontextových slov k predikci slova následujícího.

4. EXTRAKCE VLASTNOSTÍ ZAKÁZEK



Obrázek 4.18: Architektura paralelního učení modelu USE se sdíleným enkodérem[5]



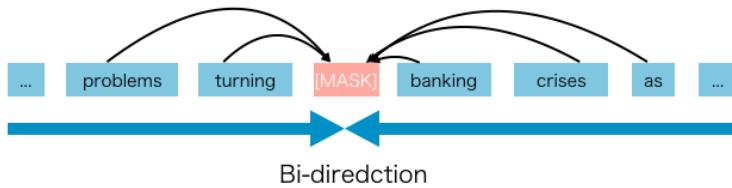
Obrázek 4.19: Směrově orientované predikce autoregresivního jazykového modelu[6]

Přístup autoregresivních jazykových modelů je ovšem omezující na dva směry, buď dopředný, nebo zpětný, jako je zobrazeno na obrázku 4.19.

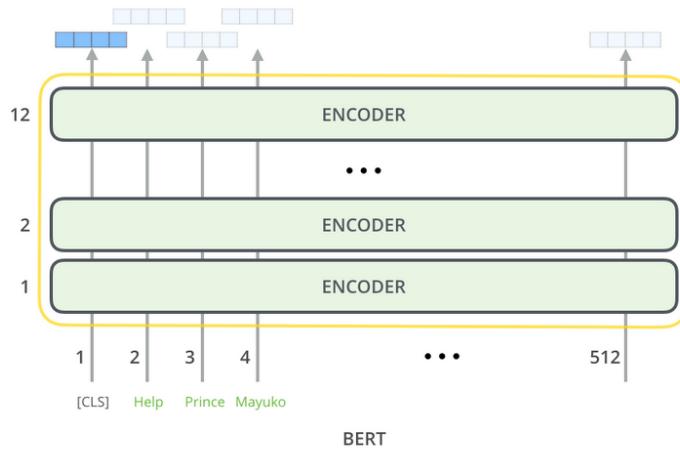
4.2.4.16 BERT

Dalším z úspěšných modelů Transformer rodiny je BERT (z angl. Bidirectional Encoder Representations from Transformers)[41], který způsobil rozruch v komunitě, kvůli dosažení state-of-the-art výsledků v mnoha různých úlohách zpracování přirozeného jazyka, říká Horev[42].

BERT je navržený pro modelování jazyka a jeho následné jednoduché ladění. Díky tomu potřebuje oproti klasickému Transformer modelu pouze mechanizmus enkodéru.



Obrázek 4.20: Obousměrně orientovaný autoenkodér[6]



Obrázek 4.21: Ilustrace architektury modelu BERT[7]

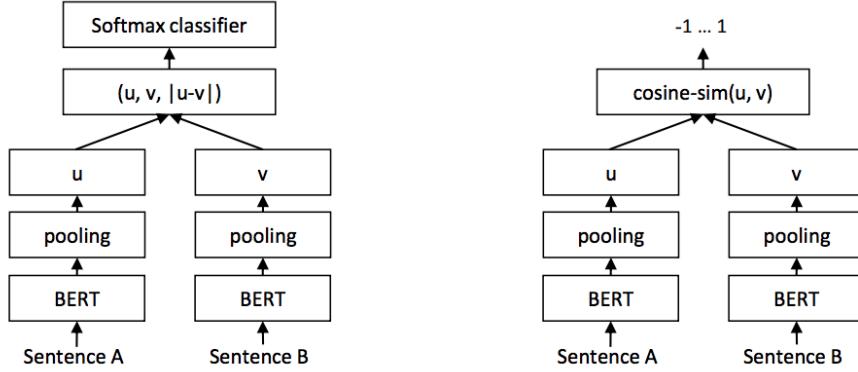
Autoři modelu přišli na nový způsob řešení predikce slova na základě kontextu. BERT nepredikuje následující slovo, ale slovo zamaskované speciální značkou. Takovému přístupu se říká autoenkódovací jazykový model (z anglicky autoencoder language model), viz. obrázek 4.20.

Této inovace BERT využívá pro komplexnější využití kontextové informace. Narození od autoregresivních modelů tak enkodér modelu BERT může číst celou sekvenci slov najednou. Konkrétně tedy nejde o dvousměrné čtení, přestože je jeho název modelu tak usouzený. To je zobrazené na obrázku 4.21.

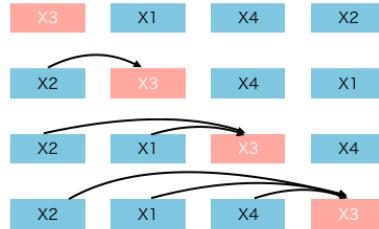
Mnohem více do detailu o architektuře Transformer a BERT popisuje Jay Alammar ve svých ilustrovaných článcích[7][43].

4.2.4.17 Sentence-BERT

Jak je vidět na obrázku 4.21, BERT doplňuje sekvenci tokenů o speciální token [CLS], kde jeho výstup lze používat pro klasifikační úlohy. Autoři modelu Sentence-BERT však ukazují[44], že poskytuje pouze slabý embedding pro jiné než klasifikační úlohy. Navrhli proto architekturu využívající BERT



Obrázek 4.22: Architektura SBERT pro trénování klasifikátoru a počítání kosinové podobnosti[3]



Obrázek 4.23: Permutační modelování jazyka[6]

jako základ pro tzv. „siamese“ a „triplet“ (zachycené na obrázku 4.22) síťovou strukturu, kterou by bylo možné zachycovat sémanticky smysluplné embeddingy vět. Tím se jim podařilo překonat dosavadní state-of-the-art metody pro embedding vět.

4.2.4.18 XLNet

XLNet[45] je model odvozený od modelu Transformer-XL[46], který vznikl jako rozšíření klasického transformeru ve smyslu zrušení omezení vstupu fixní délky. Jak Liang vysvětluje ve svém článku[6], princip autoencoderu v modelu BERT přinesl i svá omezení. Proto autoři modelu XLNet navrhli tzv. permutační modelování jazyku (z anglicky permutation language modeling). To umožňuje modelu využít kontextu z obou stran, přestože zůstává být autoregresivní, jak je naznačeno na obrázku 4.22. Tyto inovace se opět projevily pozitivně na vlastnosti modelu, díky čemuž zaujal state-of-the-art výsledky pro mnoho NLP úloh.

4.2.4.19 Další

Vývoj jazykových modelů je stále v rozmachu a tak skoro každý měsíc vychází publikace s nějakým novým objevem. V mé práci nemůžu ani zdaleka obsáhnout vše zajímavé, nicméně jeden z velice vyčerpávajících a aktuálních přehledů je například ve článku[47] od Manua Suryavanshe.

4.2.5 Metody učení s učitelem

Palachy ve svém článku[3] dále uvádí několik přístupů řešení embeddingu pomocí metod učení s učitelem. Pro účely mé práce tyto metody však nejsou důležité, proto je konkrétně nezmiňuji.

4.2.6 Výběr metody a technologie

Pro výběr vhodného modelu pro aplikaci při vyhodnocování podobnosti textů stanovuji tato kritéria. Vzhledem k doméně mé práce – české veřejné zakázky – potřebuji, aby byl model schopný pracovat s českým jazykem. Zároveň však nedisponuji prostředky k učení modelu na českém korpusu od nuly, tedy potřebuji najít před-trénovaný model podporující češtinu. Kandidátní modely poté experimentálně vyhodnocuji k finálnímu výběru toho nejhodnějšího.

4.2.6.1 Dostupné před-trénované modely

Naprostá většina různých modelů je poskytována pouze s anglickými před-trénovanými modely, kvůli čemu se pro mé účely stávají nevhodnými.

Dosavadní state-of-the-art výsledky podle portálu paperswithcode.com v kategorii sémantické podobnosti textů ukazují, že nejlépe se umíšťují modely XLNet a ALBERT[48]. Ani pro jeden z těchto modelů není publikovaný před-trénovaný model s podporou češtiny, viz. repozitář *zihangdai/xlnet*⁶ a web *Huggingface Transformers*⁷.

Dalším nadějným kandidátem je model Sentence-BERT, který sice poskytuje vícejazyčný model *distiluse-base-multilingual-cased*, avšak mezi podporovanými jazyky tohoto modelu se čeština nedostala. Model je dostupný v repozitáři autorů (*UKPLab/sentence-transformers*⁸).

Model BERT se s jeho různými mutacemi dnes těší asi největší popularitě ze všech dostupných modelů. Díky tomu je také dostupné veliké množství jeho před-trénovaných modelů. Se zaměřením na češtinu jsou dostupné modely *bert-base-multilingual-cased* (z oficiálního repozitáře⁹ autorů) či *Slavic BERT* (od skupiny *DeepPavlov*¹⁰).

⁶XLNet repozitář – <https://github.com/zihangdai/xlnet>

⁷Huggingface Transformers web – https://huggingface.co/transformers/pretrained_models.html

⁸Sentence-BERT repozitář – <https://github.com/UKPLab/sentence-transformers>

⁹BERT repozitář – <https://github.com/google-research/bert/blob/master/multilingual.md>

¹⁰DeepPavlov web – http://docs.deeppavlov.ai/en/master/features/pretrained_vectors.html

4. EXTRAKCE VLASTNOSTÍ ZAKÁZEK

Model USE disponuje vícejazyčnou verzí MUSE, publikovanou v práci „Multilingual Universal Sentence Encoder for Semantic Retrieval“[49]. Podporuje tak 16 různých jazyků, mezi kterými však čeština není.

ULMFiT se dočkal své vícejazyčné verze MultiFiT, publikované v práci[50], se kterým překonávají výsledky vícejazyčného modelu BERT. Podporuje ale jen 7 jazyků, mezi které se čeština opět nedostala.

Oproti tomu, model ELMo je vydaný s širokou podporou mnoha jazyků pod repozitářem *HIT-SCIR/ELMoForManyLangs*¹¹. Mimo jiné tak pro model ELMo existuje i česká verze.

Dalším široce rozšířeným modelem je fastText, který nabízí různé před-trénované modely pro až 157 jazyků. Všechny modely včetně českého jsou dostupné na webu *fasttext.cc*¹²

Ostatní modely jako GPT, Sent2vec, FastSent či GloVe před-trénované modely bud' vůbec nemají, nebo mají jen pro anglický jazyk. Jedině pro model GloVe existuje vícejazyčné rozšíření (dostupné na webu *GloVe*¹³), které ovšem podporu češtiny také neposkytuje.

4.2.7 Vyhodnocení modelů

Zatímco na anglický jazyk se zaměřuje po celém světě mnoho lidí, pro vyhodnocení metod embeddingů pro češtinu zatím vzniklo jen pár prací.

Jednou z nich je práce Karolny Hořeňovské – „An evaluation of Czech word embeddings“[51], kde autorka zmiňuje, že pro češtinu se mohou modely výkonnostně lišit od výsledků pro angličtinu. Uvádí to na příkladu CBOW architektury, která pro češtinu funguje lépe než skip-gram, zatímco pro angličtinu je to naopak. Navíc se v tom shoduje s předešlou prací „New word analogy corpus for exploring embeddings of Czech words“[52]. Stejně tak se shodují v tom, že GloVe model pro češtinu nefunguje dobře. Dále Hořeňovská uvádí, že i model BERT pro podobnost českých textů dosahuje překvapivě slabých výsledků.

Nejlépe se jeví model fastText, který na datasetu „Czech Dataset for Semantic Similarity and Relatedness“[53] dosahuje z testovaných modelů nejlepších výsledků.

Jedním z metodik pro vyhodnocování jazykových modelů je úloha STS (z angl. Semantic Textual Similarity), která se stala hlavní úlohou tzv. SemEval (International Workshop on Semantic Evaluation). STS úloha znamená ohodnocování podobnosti dvou textových fragmentů, přičemž výsledky STS systémů jsou porovnávány s manuálně anotovanými daty. Toto porovnávání se typicky měří korelací.

V základu jsou datasety pro STS úlohu anglické, ale v práci „Czech Dataset for Semantic Textual Similarity“[54] publikují českou verzi, jako překlad

¹¹ElMoForManyLangs repozitář - <https://github.com/HIT-SCIR/ELMoForManyLangs>

¹²FastText web - <https://fasttext.cc/docs/en/crawl-vectors.html>

¹³GloVe web - http://www.cs.cmu.edu/~afm/projects/multilingual_embeddings.html

původního anglického datasetu. Dataset je dostupný v repozitáři *Svobikl/sts-czech*¹⁴.

V rámci práce[55] vznikl nástroj pro ohodnocování kvality univerzálních reprezentací vět, tzv. *SentEval*. Tento nástroj obsahuje, mimo jiné, i úlohy STS. Své experimentální vyhodnocení modelů pro embedding textu tak zakládám na tomto nástroji, který dále upravuji pro použití českého STS datasetu.

Nástroj *SentEval* je dostupný ke stažení v repozitáři *facebookresearch/-SentEval*¹⁵. V základní verzi nepodporuje vyhodnocení embeddingů pro delší úseky textu (například celé věty), ale pouze po jednotlivých tokenech (slovách), které pouze agreguje pro získání embeddingu celého fragmentu. Dnešní modely ovšem dokázou provádět embedding pro celé věty, a proto nástroj upravuji, abych byl pomocí něj schopný vyhodnocovat embeddingy celých vět.

Vzdálenost mezi embeddingy dvou textových fragmentů každého vzorku je měřena kosinovou vzdáleností. Jako výsledek *SentEval* vrací Pearsonovy a Spearmanovy korelační koeficienty mezi podobností vypočítanou z embeddingů a manuálně anotovanou.

Pro experimentální vyhodnocení vybírám několik následujících modelů:

- *bert-base-multilingual-cased*

Vícejazyčný model typu BERT, nativně používaný NLP frameworkem *Transformers* (*huggingface/transformers*¹⁶)

- *multi_cased_L-12_H-768_A-12*

Vícejazyčný model typu BERT, dostupný z oficiálního repozitáře *google-research/bert*¹⁷, ve formě tensorflow model checkpointu

- *bg_cs_pl_ru_cased_L-12_H-768_A-12_v1*

Vícejazyčný (podporující bulharštinu, češtinu, polštinu a ruštinu) model typu BERT, dostupný z webu NLP frameworku *deeppavlov*¹⁸, ve formě tensorflow model checkpointu

- *bert-base-cased*

Anglický model typu BERT, nativně používaný NLP frameworkem *Transformers* (*huggingface/transformers*)

- *cc/cs_300_fasttext*

Český model typu fastText, dostupný z webu frameworku *fastText*, v binární formě používané knihovnou *fasttext*

¹⁴STS Czech repozitář - <https://github.com/Svobikl/sts-czech>

¹⁵SentEval repozitář - <https://github.com/facebookresearch/SentEval>

¹⁶Transformers repozitář - <https://github.com/huggingface/transformers>

¹⁷BERT repozitář - <https://github.com/google-research/bert/blob/master/multilingual.md>

¹⁸DeppPavlov web - <http://docs.deeppavlov.ai/en/master/features/models/bert.html>

4. EXTRAKCE VLASTNOSTÍ ZAKÁZEK

- *cc_en_300_fasttext*

Anglický model typu fastText, dostupný z webu frameworku *fastText*, v binární formě používané knihovnou *fasttext*

- *wiki_cs_300_fasttext*

Český model typu fastText, dostupný z webu frameworku *fastText*, v binární formě používané knihovnou *fasttext*, trénovaný na Wikipedii

- *elmo_139*

Český model typu ELMo, dostupný z repozitáře *ELMoForManyLangs*, ve formě používané framewokem *elmoformanylangs*

Dále na jednotlivých modelech provádím různé modifikace podle toho, co umožňují frameworky s nimi pracující.

Pro modely BERT, které vrací pro každý text množinu vektorů, tak v základní verzi vybírám pouze první z nich (dříve zmiňovaný [CLS] vektor), zatímco ve verzi s příponou *mean_pooled*(viz. následující tabulky) používám všechny vektory sloučené tzv. „mean pooling“ po složkách.

Vedle toho, knihovna *fasttext* umožňuje dvojí přístup k získání embeddingu, kde oba vrací odlišné vektory. První je embedding pro slovo (přípona *word*) a druhý pro celé věty (přípona *sentence*)

Při vyhodnocování provádím 4 různé testy jak pro český, tak anglický dataset a oběma přístupy vyhodnocování embeddingů podle jednotlivých tokenů (základní přístup) i celých vět (upravený přístup):

- *STSCZ*

STS úloha na českém datasetu prováděná základním přístupem nástroje *SentEval*, viz. tabulka 4.2

- *STSCZ2*

STS úloha na českém datasetu prováděná upraveným přístupem pro vyhodnocení embeddingů celých vět , viz. tabulka 4.3

- *STS16*

STS úloha na oficiálním anglickém datasetu STS16 prováděná základním přístupem nástroje *SentEval*, viz. tabulka 4.4

- *STS162*

STS úloha na oficiálním anglickém datasetu STS16 prováděná upraveným přístupem pro vyhodnocení embeddingů celých vět, viz. tabulka 4.5

Výsledky experimentů jsou zobrazené v tabulkách níže.

Na výsledcích je možné pozorovat hned několik zjištění.

Zaprvé, nejdůležitější zjištění je, že nejlépe si vede model fastText, což mimo jiné potvrzuje i tvrzení v práci Hořeňovské[51], zmíněného výše. Stejně

4.2. Embedding dokumentů

STSCZ		
Název modelu	Pearson	Spearman
<i>wiki_cs_300_fasttext_sentence</i>	0.634602	0.645182
<i>wiki_cs_300_fasttext_word</i>	0.619290	0.625226
<i>cc_cs_300_fasttext_sentence</i>	0.616563	0.624195
<i>multi_cased_L-12_H-768_A-12_mean_pooled</i>	0.536009	0.589423
<i>multi_cased_L-12_H-768_A-12</i>	0.522020	0.576311
<i>bg_cs_pl_ru_cased_L-12_H-768_A-12_v1</i>	0.520884	0.571077
<i>multi_cased_L-12_H-768_A-12_mean_pooled</i>	0.518153	0.566517
<i>elmo_139</i>	0.513847	0.531696
<i>bg_cs_pl_ru_cased_L-12_H-768_A-12_v1_mean_pooled</i>	0.496374	0.553772
<i>cc_cs_300_fasttext_word</i>	0.467745	0.486335
<i>bert-base-multilingual-cased</i>	0.424912	0.502240

Tabulka 4.2: Tabulka výsledků testu STSCZ

STSCZ2		
Název modelu	Pearson	Spearman
<i>wiki_cs_300_fasttext_sentence</i>	0.634602	0.645156
<i>cc_cs_300_fasttext_sentence</i>	0.616563	0.624252
<i>wiki_cs_300_fasttext_word</i>	0.613073	0.618057
<i>multi_cased_L-12_H-768_A-12_mean_pooled</i>	0.599592	0.612282
<i>multi_cased_L-12_H-768_A-12</i>	0.589300	0.605837
<i>bg_cs_pl_ru_cased_L-12_H-768_A-12_v1</i>	0.557063	0.577191
<i>cc_cs_300_fasttext_word</i>	0.556476	0.572473
<i>multi_cased_L-12_H-768_A-12_mean_pooled</i>	0.556184	0.573161
<i>bg_cs_pl_ru_cased_L-12_H-768_A-12_v1_mean_pooled</i>	0.536216	0.556067
<i>bert-base-multilingual-cased</i>	0.377873	0.425252
<i>elmo_139</i>	0.335470	0.347201

Tabulka 4.3: Tabulka výsledků testu STSCZ2

STS16		
Název modelu	Pearson	Spearman
<i>cc_en_300_fasttext_sentence</i>	0.493155	0.542878
<i>wiki_cs_300_fasttext_word</i>	0.421575	0.486014
<i>wiki_cs_300_fasttext_sentence</i>	0.414491	0.481977
<i>bert-base-cased</i>	0.413597	0.527323
<i>bert-base-cased_mean_pooled</i>	0.401948	0.512354
<i>cc_en_300_fasttext_word</i>	0.273957	0.345844

Tabulka 4.4: Tabulka výsledků testu STS16

STS162		
Název modelu	Pearson	Spearman
<i>bert-base-cased_mean_pooled</i>	0.645878	0.655573
<i>wiki_cs_300_fasttext_word</i>	0.563231	0.590259
<i>cc_en_300_fasttext_sentence</i>	0.493155	0.542878
<i>cc_en_300_fasttext_word</i>	0.484478	0.498747
<i>bert-base-cased</i>	0.461828	0.498818
<i>wiki_cs_300_fasttext_sentence</i>	0.414491	0.481977

 Tabulka 4.5: Tabulka výsledků testu *STS162*

tak se ukázalo, že model ELMo dosahuje obecně slabších výsledků. Nakonec model BERT dosahuje různých výsledků, závisejících na nastavení úlohy a volby konkrétního testu.

Zarážející zjištění je, že výsledky modelů *bert-base-multilingual-cased* a *multi_cased_L-12_H-768_A-12* se liší, přestože by měli být modely totožné, pouze použité jiným způsobem. Vysvětluji si to možným zveřejněním jiných verzí těchto modelů pro stažení oproti nativnímu použití frameworkm.

Stejně tak je zvláštní, že český model fastText je schopný dosahovat srovnatelných a dokonce i lepších výsledků na anglickém testu než samotné anglické modely BERT i fastText.

Změna přístupu určování embeddingu se projevila pro různé modely také různě. Zatímco u modelu fastText se téměř neprojevily, model ELMo ztratil na výkonu při embedování celé věty a model BERT zaznamenal pro některá nastavení zlepšení, ale pro některá zase zhoršení.

4.2.8 Sestavení komponenty pro extrakci embeddingu

Komponenta pro extrakci embeddingu z textu je velice jednoduchá. Přímo využívá knihovny *fasttext* pro provádění embeddingu za použití FastText modelu — konkrétně *cc_en_300_fasttext_sentence* — který se jeví podle vyhodnocení z kapitoly 4.2.7 nejlépe.

Komponenta přijímá token v podobě textového řetězce a vrací jeho embedding jako vektor reálných čísel o dimenzi 300 (defauktní nastavení modelu) v podobě *numpy*¹⁹ pole.

Implementace komponenty je adaptovaná pro příjem kolekce tokenů, kdy vrací kolekci obsahující embedding pro každý z tokenů.

Pro inicializaci vyžaduje předání binárního *fasttext* modelu, nebo alespoň systémové cesty k němu.

¹⁹NumPy – Python knihovna pro vědecké výpočty;
dostupná z <https://pypi.org/project/numpy/>

4.3 Extrakce předmětu

V této kapitole popisuji proces extrakce předmětu z dokumentace veřejných zakázek (viz. kapitola 3.1).

Extrakce předmětu ve své podstatě zastává stěžejní část předzpracování dat celého systému. Jejím cílem je vyextrahat předmětnou informaci z nestrukтуrovaného textu celé dokumentace VZ. Přestože jsou data dokumentace VZ nestrukturovaná, jde o poměrně specifický typ dat, ve kterém se zpravidla opakují určité vzorce. Tento proces se snaží využít těchto vzorců k předzpracování textu, primárně spočívajícího v čistění dat v podobě různého filtrování, transformací a extrakcí.

4.3.1 Návrh procesu extrakce předmětu

Při tzv. „black-box“ pohledu na proces je na vstupu veškerá dostupná dokumentace veřejné zakázky a na výstupu výčet konkrétních předmětných částí této VZ.

Pro takovou transformaci navrhoji proces sestávající z těchto hlavních kroků (podprocesů):

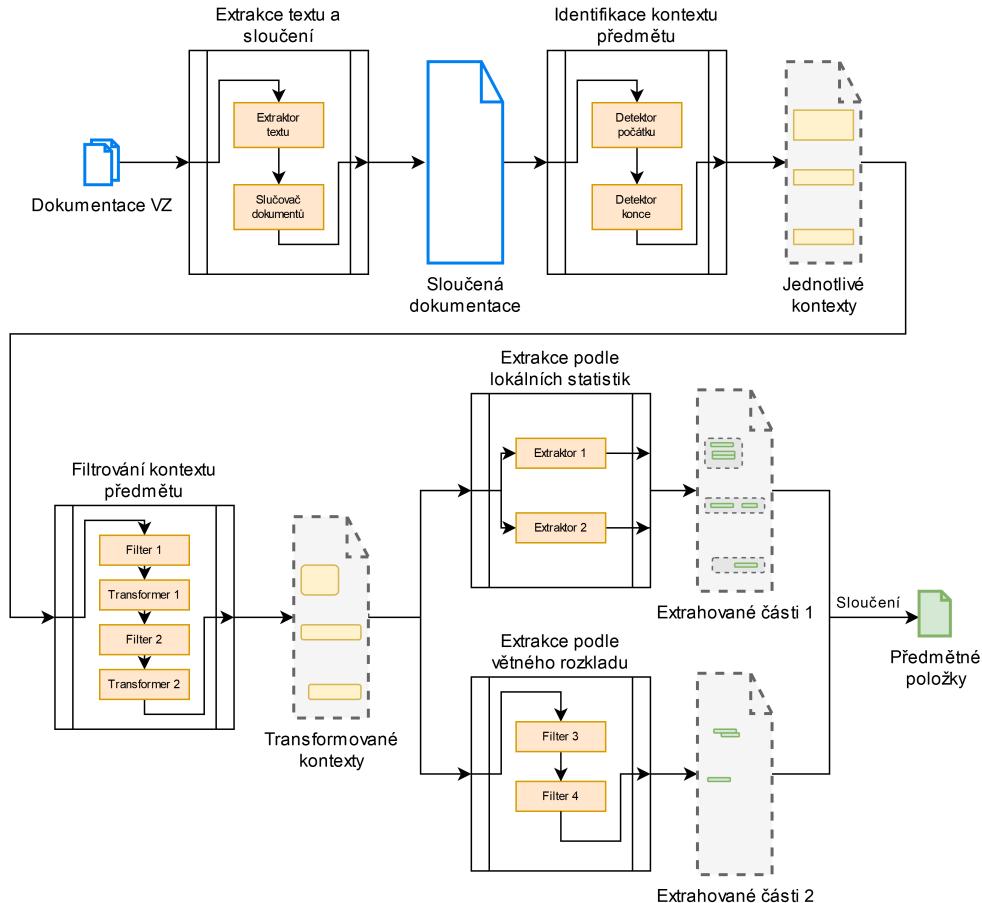
1. Extrakce textu z dokumentů
2. Identifikace kontextu předmětu v dokumentech
3. Filtrování a transformace kontextu
4. Extrakce předmětných částí podle lokálních charakteristik
5. Extrakce předmětných částí podle větného rozkladu

V textu dále používám specifické názvosloví pro symbolické odlišení druhu procesů, které provádí operace s částmi textu:

- filter:= odfiltrovává; část textu, která vyhovuje podmínce, je odstraněna,
- transformer:= transformuje; část textu, která vyhovuje podmínce, je nahrazena definovanou či odvozenou náhradou,
- extraktor:= extrahuje; část textu, která vyhovuje podmínce, je ponechána.

Celý proces je schématicky znázorněn na obrázku 4.24.

4. EXTRAKCE VLASTNOSTÍ ZAKÁZEK



Obrázek 4.24: Schéma procesu extrakce předmětu dokumentace VZ

4.3.2 Extraktce textu z dokumentů

Jak napovídá schéma 4.24, na vstupu procesu extrakce textu je veškerá dostupná dokumentace k VZ v podobě tzv. „formátovaného“ textu (z angl. rich-text). Na výstupu se poté očekává souvislý „prostý“ text (z angl. plain-text).

V dokumentaci VZ se standarně vyskytují předsmluvní dokumenty, přílohy (projektové dokumentace, specifikace), smlouvy a mnoho dalšího. Z většiny se jedná o textové dokumenty ve formátu *.pdf* či *.doc/.odt*, ale vyskytují se i další od tabulek (*.xls*), přes obrázky (*.jpg*) až po komplexní archivy (*.zip*) nebo certifikáty (*.cer*). Ze všech těchto formátů potřebuji extrahouvat prostý text.

K tomu využívám nástroje projektu *public-contracts*, který je jako open-source dostupný v repozitáři *opendatalabcz/public-contracts*²⁰, vyvýjeného v

²⁰ opendatalabcz/public-contracts repozitář - <https://github.com/opendatalabcz/public-contracts>

rámci laboratoře otevřených dat *OpenDataLab*²¹, který používá pro extrakci textu nástroje *Apache Tika* s OCR technologií *Tesseract*.

Samotný projekt *public-contracts* řeší vyhledávání českých VZ, stejně jako stahování jejich dokumentace, extrakci textu z nich a následné uložení do databáze. Implementován je v jazyku Java a pro ukládání vyextrahovaných textů používá databázi Postgres.

Takto vyextrahovaný text nakonec slučuje pro použití v dalším procesu.

4.3.3 Identifikace kontextu předmětu

Některé zakázky obsahují rozsáhlé projektové dokumentace o desítkách až stovkách stran. Kvůli tomu je nutné takový rozsah omezit na kontext, ve kterém se pojednává o samotném předmětu. Zároveň je výběr správné části dokumentu přínosný i obecně pro zpřesnění výsledků, díky odfiltrování velikého množství nerelevantní informace z textu.

Proces identifikace kontextu předmětu tak provádí transformaci souvislého prostého textu na jeho podmnožiny, jak je znázorněno na schématu 4.24.

Slovo kontext předmětu zde ideálně znamená část textu obsahující samotnou kapitolu předmětu, což je obvykle rozsah od názvu kapitoly po její konec.

4.3.3.1 Určení počátku kontextu

Určení počátku kontextu vypočítávám ve čtyřech krocích:

1. Nalezení klíčových slov

Naprostá většina dokumentace VZ pojednává o předmětu v kapitole s názvem odvozeným od slova „předmět“, přičemž v samotné kapitole se poté různě vyskloňované slovo často dále objevuje. Na základě toho specifikují množinu klíčových slov, které v celém textu vyhledávám. Vzhledem k tomu, že klíčová slova se mohou vyskytovat mnohonásobně, dochází k falešným nálezcům. Kvůli tomu jednotlivá klíčová slova ohodnocuju body, které vyjadřují sílu jejich výskytu k určení počátku kontextu.

Klíčová slova a jejich ohodnocení mohou být například následující:

Klíčové slovo	Ohodnocení
Předmět smlouvy	10
Předmět plnění	10
Název veřejné zakázky	3
Předmět	1
Popis	1

Tabulka 4.6: Klíčová slova a jejich ohodnocení

²¹OpenDataLab web - <https://opendatalab.cz/>

4. EXTRAKCE VLASTNOSTÍ ZAKÁZEK

Označení	Charakteristika	Úprava koeficientu
CH1	Výskyt tvoří celou samostatnou řádku	+2
CH2	Přesný výskyt (odpovídají velká/malá písmena)	+1,5
CH3	Výskyt psaný verzálkami	+1,5
CH4	Blízké odrádkování za výskytem	+2
CH5	Číslování předcházející výskytu	+2
CH6	Římské číslování předcházející výskytu	+2
CH7	Blízký výskyt slovesa „je“ za výskytem ²²	+2
CH8	Slovo „článek“ předcházející výskytu	+2
CH9	Blízký výskyt jména „Zboží“ za výskytem ²³	-1
CH10	Běžná věta za výskytem ²⁴	+ poměr malých písmen

Tabulka 4.7: Detekované charakteristiky a jejich dopad na multiplikativní koeficient ohodnocení klíčových slov

Ne všechny VZ obsahují ve své dokumentaci kapitolu s předmětem. Proto specifikuj i další obecná slova jako „název“ či „popis“ pro odchycení obecnějších vzorů.

2. Identifikování lokálních formátovacích charakteristik jednotlivých výskytů

Přestože se převedením dokumentů na prostý text ztratí mnoho informace o formátování, některé lokální formátovací charakteristiky jsou uchovány i v prostém textu. Takovými jsou například: velká písmena, číslování kapitol, ale i jen obyčejné odrádkování.

Těchto charakteristik se snažím využít v podobě multiplikativního koeficientu na body klíčového slova. Pro každý výskyt tak detekuju jednotlivé charakteristiky a při pozitivní detekci patřičně upravuji koeficient, který je základem rovný jedné.

Charakteristiky, které detekuju jsou zaznamenány v tabulce 4.7.

Při zachování klíčových slov z příkladu tabulky 4.6 je výpočet jejich ohodnocení znázorněn na obrázku 4.25.

3. Výběr nejlepších kandidátů

Pomocí samotného ohodnocení výskytů s ohledem na využití lokálních charakteristik je možné nalézt lokální maxima (např. nadpisy kapitol) avšak stále může docházet k falešným výskytům z textu odstavce (např. kumulací výskytů v odstavci). Ke zmírnění takového jevu implementuji

²²Sloveso „je“ se často vyskytuje v první větě kapitoly předmětu

²³Casto se v kapitole používá zástupných názvů pro dílo či zboží, čímž ztrácí relevantní informaci

²⁴Běžná věta obsahuje poměrně hodně malých písmen oproti ostatním znakům

...
 1000: dle přílohy č.1.
 1017:
 1019:
 1021: 2. PREDMĚT PLNĚNÍ VEŘEJNÉ ZAKÁZKY
 1057: Předmětem plnění této veřejné zakázky je dodání zemědělských strojů,
 1126: vyhovujících popisu v příloze č.2. "Specifikace zboží"
 1183: Prodávající se zavažuje (v souladu s § 409 obchodního zákoníku)
 ...

Označení	Nález 1 [1025]	Nález 2 [1025]	Nález 3 [1057]	Nález 4 [1138]
Základ	1	1	1	1
CH1	-	-	-	-
CH2	-	-	+1,5	-
CH3	+1,5	+1,5	-	-
CH4	-	+2	-	-
CH5	+2	+2	-	-
CH6	-	-	-	-
CH7	-	-	-	-
CH8	-	-	-	-
CH9	-	-	-	-
CH10	+0,83	+0,86	+0,86	+0,76
Ohodnocení	1	10	1	1
Celkem	5,33	73,6	3,36	1,76

Obrázek 4.25: Příklad výpočtu ohodnocení výskytů klíčových slov

Na příkladu jsou nalezeny dva nálezy klíčového slova „Předmět“ a jeden nález slova „Předmět plnění“. Jsou detekovány charakteristiky přesného výskytu (CH2), verzálky (CH3), odřádkování (CH4) a číslování (CH5). V odstavci následuje běžná věta (CH10), což také v kladném smyslu ovlivňuje výsledné ohodnocení.

algoritmus pro selekci nejlepšího možného výskytu na základě globálních statistik výskytů.

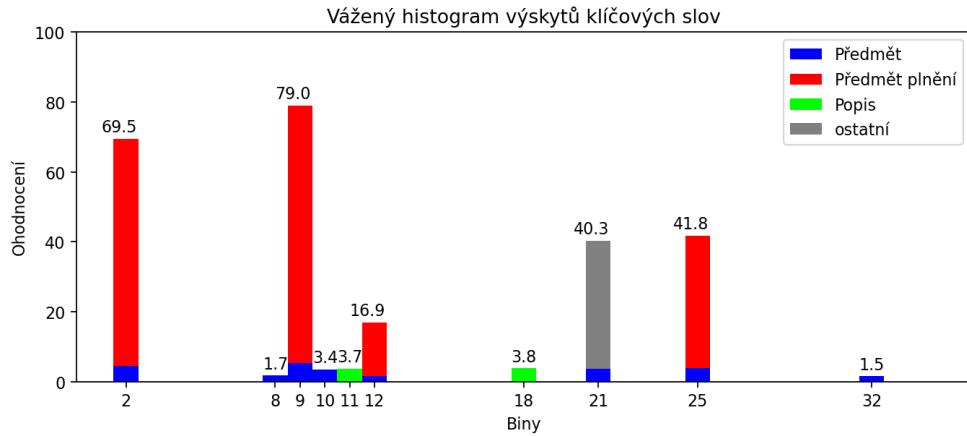
Tento algoritmus spočívá v diskretizaci pomocí biningu a následné aplikace konvoluce s jádrem potlačujícím nežádoucí jevy.

Diskretizaci provádí rozdelením celého textu na biny o stejně šířce, kterou odvozuji ze základní délky kontextu jako jeho třetinu (řádově tisíce znaků; defaultně 2040²⁵: šíře binu je poté 680). Na tomto diskrétním prostoru počítám vážený histogram ohodnocení výskytů klíčových slov spadajících do jednotlivých binů.

Následuje krok konvoluce, který z ohodnocení jednotlivých binů specifickým způsobem agreguje okolní biny jako skóre daného binu. Konvoluční jádro stanovuji na šestici (1, 1, 2, -2, -1, -1), která má za cíl určit samotný počátek kapitoly týkající se předmětu. Část (-2, -1, -1) penalizuje hodnoty tří předcházejících binů, zatímco část (1, 1, 2) aggreguje

²⁵Defaultní délku kontextu odvozuji expertním odhadem rozsahu kapitoly v počtu znaků (nízké tisíce) a zároveň s ohledem na dělitelnost velikostí konvolučního jádra (pro celočíselné rozdělování do binů)

4. EXTRAKCE VLASTNOSTÍ ZAKÁZEK



Obrázek 4.26: Histogram výskytů klíčových slov pro výběr kontextu
Biny jsou pro ilustraci o šíři 100 znaků (rozsah kontextu – 300 znaků). Úryvek z příkladu 4.25 je zde zastoupen biny 9 a 10, přičemž bin 9 nabývá nejvyššího ohodnocení.

tři následující s dvojnásobným umocněním prvního binu na pomyslném rozhraní kapitol.

Některé VZ mají celý předmět rozdělen do několika podobně strukturovaných kapitol. Kvůli tomu nevybírám pouze bin s nejvyšším skóre, ale vybírám všechny biny, které dosahují alespoň dvou třetin maximálního dosaženého skóre.

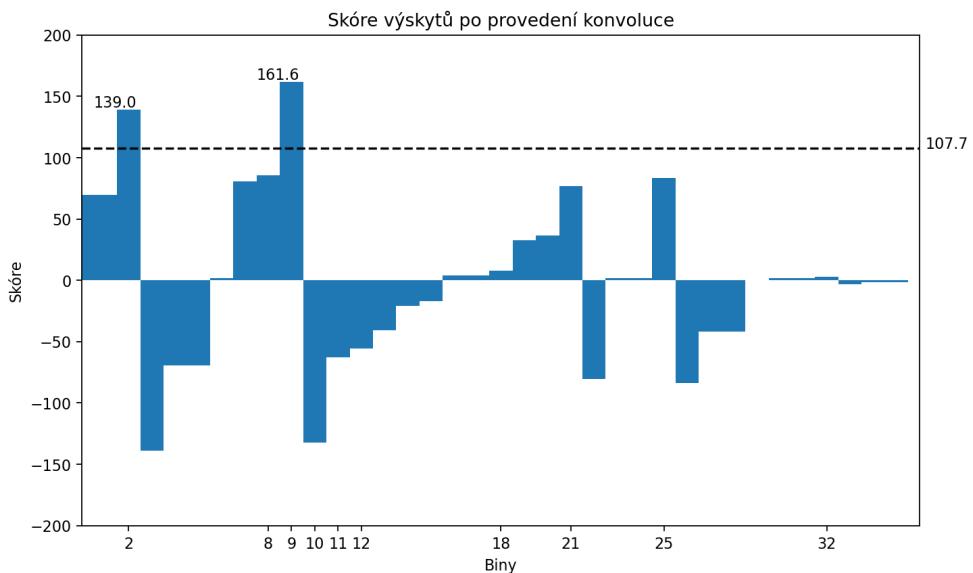
Navázáním na příklad z tabulky 4.6 a obrázku 4.25 dále znázorňuji na obrázku 4.26 výpočet histogramu. Výsledek aplikace konvoluce na histogram 4.26 je dále zobrazen na následujícím diagramu 4.27.

4. Korekce samotného počátku kontextu

Po vzoru kroku 1. opakuji hledání výskytů těch samých klíčových slov v hrubě určených kontextech z předešlých kroků. Z těchto výskytů vybírám jeden nejvýznamnější, který se stává výsledným počátkem daného kontextu.

4.3.3.2 Určení konce kontextu

Pouhý odhad ukončení kontextu na základě nastavené délky rozsahu je velice hrubá metoda, která v kontextu může zanechávat nežádoucí informace. Pokud však víme přesný počátek, můžeme na základě lokálního formátování takového počátku usuzovat, kde začíná další kapitola a stejně tak i určit, kde končí kontext předmětu.



Obrázek 4.27: Znázornění výběru nejlepších kandidátů na kontext podle skóre. Ohodnocení vnitřních binů kapitoly jsou potlačené (biny 10, 11, 12), zatímco v binu 9 jsou jejich výskyty agregovány a je zvýrazněna jeho důležitost pro výběr počátku kontextu. Horizontální čára znázorňuje rozhraní pro výsledný výběr binů 2 a 9.

Následující detektory implementují pro podchycení lokálního formátování počátků kapitol:

1. Číslování kapitol

Detektor detekuje hierarchické číslování kapitol na základě regulárního výrazu. Pokud je číslování detekováno, vypočítám číslo následující kapitoly, které dále detekuju v kontextu. Pokud dojde k pozitivnímu nálezu, určím jeho pozici ukončení kontextu.

2. Římské číslování kapitol

Obdobný detektor, který místo klasických čísel pracuje s římskými čísly.

3. Kapitoly uvedené klíčovým slovem

Jednoduchý detektor, který hledá klíčové slovo „článek“ před počátkem kontextu. Pokud je slovo detekováno, je opět hledáno v kontextu a v případě nálezu určuje jeho ukončení.

4. Název kapitoly psaný verzálkami

Detektor detekuje, zdali je první řádek kontextu psaný velkými písmeny. Pokud ano, hledá další takový řádek a určí ho koncem kontextu.

4. EXTRAKCE VLASTNOSTÍ ZAKÁZEK

5. Pasáž uvedená klíčovým slovem

V tomto případě nejde o určování konce kapitoly, nýbrž o ukončení předmětné fráze. Detektor hledá klíčové slovo „název“ kolem počáteční pozice, které předznamenává, že nejde o samostatnou kapitolu, ale jen o frázi s názvem předmětu. V takovém případě ukončuje kontext s prvním odřádkováním za počátkem.

6. Fráze v uvozovkách

Pokud jsou detekovány za počátkem kontextu otevírací uvozovky, hledá detektor k nim do páru uzavírací symbol. Při nálezu je s ním kontext ukončen.

7. Zakázaná klíčová slova

Dokumentace VZ často kopírují podobnou strukturu. V mnoha případech se tak za kapitolou pojednávající předmět řeší cena, doba, či místo plnění. Na základě toho detektor detekuje klíčová slova: „cena“, „doba“ a „místo“ a jejich případným nálezem ukončuje kontext.

8. Defaultní odhad délky

V případě selhání všech předešlých detektorů zůstává kontext ukončený nastavenou maximální délkou.

Pro příklad nálezu kontextu v úryvku 4.25 by se aplikovaly detektory 1 a případně 4.

4.3.3.3 Měření dopadu jednotlivých algoritmů a jejich parametrů

Měření provádím oproti množině 40 pseudo-náhodně vybraných veřejných zakázek, u kterých jsem manuálně validoval výstup extrakce. Jako metriku pro podobnost vyextrahovaného a referenčního textu používám Jaccard index (viz. kapitola 5.1).

Pro každou VZ aplikují tuto metriku a nakonec počítám výsledné skóre jako jejich průměr.

V následujících tabulkách jsou zaznamenány výsledky měření pro jednotlivá klíčová slova, lokální charakteristiky a detektory. Uvedené doby běhu jednotlivých měření jsou pouze orientační. Vždy provádím měření pouze „s“ danou entitou a „bez“ ní (se všemi ostatními).

V tabulce 4.8 můžeme vidět, jaký dopad na přesnost extrakce předmětu mají jednotlivá klíčová slova. Je vidět, že slovo „předmět“ a „popis“ zatěžují extrakci nejvíce, což bude zřejmě přímo implikováno počtem jejich výskytů. Z měření je dále zřejmé očekávané chování slova „předmět“, které samo o sobě dokáže do jisté míry podchytit kontext, zatímco na jeho přesné určení má malý vliv. Oproti tomu, sousloví „předmět smlouvy“ má nejznatelnější dopad na upřesnění kontextu.

4.3. Extrakce předmětu

Klíčové slovo (ohodnocení)	Pouze jedno		Bez jednoho	
	Skóre	Čas [s]	Skóre	Čas [s]
Vše	100.00	12.1	-	-
Předmět smlouvy (10)	44.98	1.4	68.01	11.5
Předmět díla (10)	7.52	0.5	94.97	11.5
Předmět plnění (10)	14.81	1.8	88.50	11.9
Předmět veřejné zakázky (10)	20.32	0.7	89.02	13.1
Vymezení předmětu (10)	8.76	0.4	93.84	12.6
Vymezení plnění (10)	2.61	0.5	97.57	12.5
Popis předmětu (10)	12.52	0.7	97.05	12.5
Název veřejné zakázky (3)	12.88	2.0	92.28	11.2
Veřejná zakázka (1)	8.79	3.2	98.46	11.2
Veřejné zakázce (1)	13.22	1.6	96.02	11.0
Předmět (1)	66.11	7.0	97.51	6.4
Popis (1)	10.64	4.1	98.96	8.4

Tabulka 4.8: Měření dopadu jednotlivých klíčových slov na extrakci

Charakteristika	Pouze jedna		Bez jedné	
	Skóre	Čas [s]	Skóre	Čas [s]
Základ/Vše	64.48	12.9	100.00	10.6
CH1	78.95	12.0	96.53	11.6
CH2	77.30	12.2	97.13	11.9
CH3	64.82	12.2	98.73	11.4
CH4	77.72	11.9	98.73	11.4
CH5	65.21	11.3	89.89	12.1
CH6	72.12	11.6	94.79	11.4
CH7	64.25	11.9	98.60	11.8
CH8	72.13	11.9	97.76	11.5
CH9	64.75	11.2	99.08	12.2
CH10	62.71	12.1	97.70	11.1

Tabulka 4.9: Měření dopadu jednotlivých lokálních charakteristik na extrakci

V tabulce 4.9 je vidět dopad jednotlivých lokálních charakteristik pro úpravu koeficientu ohodnocení výskytů klíčových slov. Ani jedna charakteristika nemá obzvlášť citelný dopad na výpočetní náročnost. Sama o sobě má největší vliv CH1 – nález obsahuje samostatnou rádku, zatímco největší dopad na upřesnění kontextu má potom CH5 – číslování kapitol (zároveň i CH6 – římské číslování má druhý největší dopad).

Nakonec v tabulce 4.10 můžeme pozorovat dopad detektorů ukončení kontextu. Nejvýznamnější dopad mají detektory 1 a 2 (detektory klasického respektive římského číslování kapitol. Dobře také funguje detektor 7 – ukončovací

4. EXTRAKCE VLASTNOSTÍ ZAKÁZEK

Detektor	Pouze jeden		Bez jednoho	
	Skóre	Čas [s]	Skóre	Čas [s]
Základ/Vše	34.14	11.0	100.00	12.1
1	58.40	11.6	83.73	12.5
2	58.40	11.6	90.96	12.3
3	39.30	11.6	97.83	12.4
4	40.88	12.6	98.71	12.2
5	40.81	12.0	93.83	12.2
6	36.47	11.9	97.67	12.4
7	61.64	12.2	91.69	12.2

Tabulka 4.10: Měření dopadu jednotlivých detektorů na extrakci

klíčová slova. V tabulce lze pozorovat zvláštní jev u prvních dvou detektorů, které mají stejné skóre i čas. Podle mého úsudku je to způsobené chybou v měření.

4.3.4 Filtrování a transformace kontextu předmětu

Veliké množství dokumentace veřejných zakázek je uveřejněno ve formátu oskenovaných papírových dokumentů. Z takových dokumentů extrahuji text pomocí OCR technologie, což do prostého textu zanáší různý šum, jako například záhlaví/zápatí stran, špatné odřádkování a členění textu, či chybně rozpoznané znaky.

Kvůli tomu je potřeba před samotnou extrakcí předmětných částí provést různá filtrování, transformace či případné korekce daného kontextu předmětu. Tento proces tak vstupní kontext v původní podobě zpracovává do čisté podoby.

Jak je vidět na schéma 4.24, proces sériově aplikuje jednotlivé procesory (filtery/transformery), kde výstup jednoho přichází jako vstup do následujícího. Jelikož jsou některé procesory závislé na předchozích, záleží na pořadí jejich aplikace.

Účel mnoha procesorů je intuitivní, zatímco některých tolík ne. To může být způsobeno tím, že některé procesory implementují za účelem přípravy vstupu pro algoritmy extrakce a tagovací model, který je citlivý na čistotu textu. Tento model používám dále při extrakci podle větného rozboru vět (více viz. kapitola 4.3.6).

Dále popisují jednotlivé procesory v pořadí, v jakém je aplikují.

1. Odstranění číselných řádků

Řádky s nadpolovičním poměrem číselných znaků jsou pravděpodobně řádky obsahující nerelevantní text (např. identifikátory, ceny).

2. Odstranění příliš krátkých řádků

Neprázdné řádky kratší šesti znaků jsou pravděpodobně jen pozůstatky formátovaného textu a nenesou žádnou hodnotnou informaci.

3. Odstranění nerelevantních řádků

Za použití klíčových slov a regulárních výrazů definuji několik podmínek charakterizující nerelevantní řádky, jako jsou:

- stránkování,
- telefonní čísla a fax,
- obchodní identifikátory,
- webové a emailové adresy.

4. Korekce interpunkce na konci vět

Pomocí regulárního výrazu detekuju chybné čárky na místech, kde je očekávaná tečka.

5. Odstranění číslování odstavců

Opět za použití regulárního výrazu odmazávám z počátků řádu číslování.

6. Filter prázdných řádků

Na základě délky řádků implementuji algoritmus odstranění zavádějícího odřádkování způsobeného chybou extrakcí textu. Více v kapitole 4.3.4.1.

7. Sjednocení symbolů pro uvozovky

Nahrazení českých znaků pro uvozovky univerzálním znakem „.

8. Transformace příliš dlouhých řádků

Na základě výskytu specifických znaků a délky řádků transformuji vybrané řádky do několika kratších. Více v kapitole 4.3.4.2

9. Doplnění teček na konce řádků

Jeden z transformerů implementovaných z důvodů přípravy textu pro tagovací model.

10. Doplnění dvojteček

Transformer připravující text pro extrakci podle lokálních statistik. Detekuje situaci, kde se vyskytuje klíčové slovo „název“ či „popis“ před názvem entity s počátečním velkým písmenem. V takové situaci doplňuje před název dvojtečku.

11. Nahrazení dvojtečky za dvojici „;“

Další transformer připravující text pro tagovací model.

12. Nahrazení vícenásobných teček za jednu.

Obdobně jako předchozí transformér.

13. Odstranění textu v závorkách

V závorkách je obvykle uvedena pouze dodatečná informace, která zpravidla není předmětná. Pomocí regulárního výrazu tak závorky odstraňují i s textem, který obsahují.

4.3.4.1 Filter prázdných řádků

U některých dokumentů dochází při extrakci textu ke špatně rozpoznanému odřádkování. V extrémních případech až do té míry, že jednotlivé řádky jsou oddělené několika odřádkováním, přestože text souvisle navazuje ve větách. Na druhou stranu, odřádkování obecně mají v textu svůj význam a i v případě extrakce textu mohou přinášet užitečnou informaci (více v kapitole 4.3.5).

Pro zaručení správného fungování extraktorů je nutné potlačit jev falešných odřádkování. To zajišťuje tento filter, kterým se v ideálním případě snažím dosáhnout toho, aby souvislá věta nebyla vůbec rozdělena odřádkováním.

Nejdříve je potřeba detektovat, zdali se v textu vyskytuje nějaká falešná odřádkování a následně tato odřádkování potlačit.

Pro detekci výskytu falešných odřádkování v textu implementuji algoritmus založený na globální statistice délek řádků. Myšlenka algoritmu spočívá v tom, že dokument bez falešných odřádkování má veliký rozptyl délek řádků, zatímco dokument, který má falešná odřádkování (např. odřádkování podle šíře listu papíru) má tento rozptyl malý. Algoritmus se skládá z následujících čtyř kroků:

1. Určím délku plného řádku v počtu znaků. Kvůli potlačení extrémů nevybíram maximální délku, ale délku řádku odpovídající kvantilu = 0,95 (do množiny nezapočítávám prázdné řádky).
2. Vzhledem k tomu, že v dokumentaci se běžně používá proporcionálního písma, plné řádky mohou dosahovat znatelně odlišných délek. Takovou odchylku definuji expertním odhadem na 15 %, podle čeho určuji rozhraní pro výběr všech „plných“ řádků jako alespoň 85 % délky určené v předchozím kroku.
3. Pokud je množina „plných“ řádků dostatečně veliká, odebíram maximální a minimální prvek pro stabilnější statistiku. U výsledné množiny počítám standardní odchylku jejich délek.
4. Pokud je tato odchylka menší než nastavená hranice (defaultně 5), detekuju text jako „obsahující falešné řádky“ a je dále zpracován pro jejich potlačení.

Oproti algoritmu detekce výskytu falešných odřádkování, algoritmus na jejich samotné potlačení je založený na lokální statistice délek řádků se základní myšlenkou takovou, že pokud se za sebou vyskytuje několik „plných“ řádků, odřádkování mezi nimi jsou falešná a je žádoucí je odstranit.

Výčet podmínek pro zrušení (falešných) odřádkování je následující:

1. pokud poslední řádka²⁶ je zároveň „plná“ a tvoří běžnou větu (má poměr malých písmen větší než určená hranice - defaultně 0,6), nebo
2. pokud poslední řádka nekončí ukončením věty (tečka na jejím konci) a zároveň jí předcházející řádka je „plná“, nebo
3. pokud následující řádka je „plná“ a nezačíná novou větou (velké písmeno na začátku),
4. nakonec nahrazuji všechna vícenásobná odřádkování právě dvěma.

Těmito podmínkami jsem schopen podchytit většinovou část falešných odřádkování, a to i u takových řádek, které ruší souvislý blok „plných“ řádek tím, že sama není „plná“.

Pro příklad uvádím na obrázku 4.28 ilustraci algoritmu na úryvku textu. Na obrázku 4.29 je poté znázorněn výsledný text po aplikaci filteru, kde je vidět, že zbyly pouze nefalešné odřádkování.

4.3.4.2 Transformer příliš dlouhých řádků

U některých řádků dochází k jevu, že tvoří dlouhý souvislý řetězec, ačkoli by mohly být rozděleny do více kratších řádků. Například řádek obsahující více pomlček je vhodné rozdělit do několika bodově strukturovaných řádků.

Takové rozdělování řeší tento transformer, který detekuje znaky tvořící odrážkovou/výčtovou strukturu a následně do takové struktury člení řádky, ve kterých se detekované znaky vyskytují.

Implementuji algoritmus, který nejprve detekuje znaky tvořící výčtovou strukturu a poté rozděluje řádky je obsahující. Kromě tzv. „bílých znaků“ počítám výskytu prvních znaků všech řádek. Z napočítané množiny vybírám nejpočetnější nealfanumerický znak a podle něj rozděluji řádky, ve kterých se vyskytuje (kromě výskytů v uvozovkách).

4.3.4.3 Měření dopadu jednotlivých filterů a transformerů

Podobně jako u procesu extrakce kontextu provádí měření dopadu jednotlivých filterů a transformerů na extrakci předmětu. Používám stejnou množinu dokumentů i metriku. Jako referenční data používám manuálně ověřenou a opravenou množinu výstupu celého procesu extrakce předmětu.

²⁶Poslední řádkou je myšlena poslední neprázdná řádka včetně řádky aktuálně řešené, nebo řádka prázdná, ovšem rozpoznána jako nefalešná.

4. EXTRAKCE VLASTNOSTÍ ZAKÁZEK

...
sjednanou touto Smlouvou. ●
● 4
Seznam Zboží vč. požadavku na množství, místo plnění (dodání) a kontaktní osoby odpovědné za převzetí Zboží je uveden v příloze č. 1 této Smlouvy (Seznam Zboží ● 2 ● 2 ● 2 vč. místa dodání). ●
● 4
Detailní specifikace veškerých druhů Zboží, které mohou být poptávány v rámci dynamického nákupního systému, vč. Zboží, které je předmětem této Smlouvy (identifikace dle kódu a názvu typové položky), je uvedena v příloze č. 2 této Smlouvy (Specifikace Zboží - bílé ● 1 ● 1 a další zboží). ●

77,3
0,85Q_{0,95}
90,95
Q_{0,95}

Obrázek 4.28: Ilustrace algoritmu filtru prázdných řádek
Falešná a nefalešná odřádkování jsou označena červenými, respektive zelenými body. Je vidět detekce „plných“ řádek jako řádek delších 77 znaků. Modrými čísly je nakonec naznačeno, která podmínka se uplatňuje pro zrušení odřádkování.

...
sjednanou touto Smlouvou. ●
●
Seznam Zboží vč. požadavku na množství, místo plnění (dodání) a kontaktní osoby odpovědné za převzetí Zboží je uveden v příloze č. 1 této Smlouvy (Seznam Zboží vč. místa dodání). ●
●
Detailní specifikace veškerých druhů Zboží, které mohou být poptávány v rámci dynamického nákupního systému, vč. Zboží, které je předmětem této Smlouvy (identifikace dle kódu a názvu typové položky), je uvedena v příloze č. 2 této Smlouvy (Specifikace Zboží - bílé a další zboží). ●

Obrázek 4.29: Ilustrace výsledku algoritmu filtru prázdných řádek

Procesor	Pouze jeden		Bez jednoho	
	Skóre	Čas [s]	Skóre	Čas [s]
Základ/Vše	35.57	38.1	78.53	35.2
1 Odstranění číselných řádků	35.32	36.5	75.73	36.9
2 Odstranění příliš krátkých řádků	39.20	36.6	72.92	35.7
3a Odstranění stránkování	36.08	40.3	76.52	35.0
3b Odstranění telefonních čísel	35.40	38.3	77.71	34.4
3c Odstranění obchodních identifikátorů	35.66	38.4	78.53	34.5
3d Odstranění webových adres	35.58	37.0	78.02	34.3
4 Korekce interpunkce	35.65	39.4	77.59	34.3
5 Odstranění číslování odstavců	36.93	40.1	78.09	34.7
6 Filter prázdných řádků	37.52	42.5	55.51	34.1
7 Sjednocení uvozovek	45.53	41.6	63.79	34.4
8 Transformace příliš dlouhých řádků	35.50	39.5	75.07	35.4
9 Doplnění teček	33.30	41.5	75.87	34.7
10 Doplnění dvojteček	35.72	41.7	78.36	34.7
11 Nahrazení dvojteček	43.56	40.7	75.36	34.7
12 Nahrazení teček	43.56	37.2	77.71	35.0
13 Odstranění závorek	37.64	37.2	74.11	35.6

Tabulka 4.11: Měření dopadu jednotlivých filterů a transformerů na extrakci předmětných položek

V tabulce 4.11 je vidět výsledky měření. Změny časů napovídají, že některé procesory urychlují výslednou dobu běhu procesu extrakce. Zároveň je také vidět, že téměř všechny procesory mají vliv na přesnost extrakce. Jediný procesor, který nezlepšuje skóre je procesor 3c, který ve validační množině dokumentů zřejmě nenašel uplatnění.

4.3.5 Extrakce podle lokálních charakteristik

V předešlých kapitolách jsem narážel na fakt, že přestože je většina formátování zdrojového textu extrahováním ztracena, některé charakteristiky zůstávají a lze je využít. Přesně o to se snažím tímto procesem, jak dále popisuji v této kapitole.

Položky předmětu bývají často v dokumentaci vyjmenované v seznamech strukturovaných odrážkami či odřádkovánimi. Takové seznamy se snažím pomocí následujících extraktorů detektovat a extrahovat. Kromě seznamů se dále zaměřuji na název zakázky, který často sám o sobě nese předmětnou informaci. Výstupem tohoto procesu tak je výčet konkrétních předmětných položek a názvů.

Extraktory sloužící k účelům tohoto procesu jsou následující:

1. Výčet položek podle struktury odřádkování

4. EXTRAKCE VLASTNOSTÍ ZAKÁZEK

Extraktor detekuje vzor v odřádkování a podle takového vzoru vrací výčet položek (více viz. kapitolu 4.3.5.1).

2. Výčet položek podle odrážek

Extraktor využívá podobný algoritmus detekce znaků tvořících odrážky jako transformer příliš dlouhých řádků (viz. kapitola 4.3.4.2) a podle těchto odrážek extrahuje položky. Více do detailu popisují v samostatné kapitole 4.3.5.2.

3. Rozšíření výčtu položek

Zde najde o extraktor samotný, ale o rozšíření dvou předešlých. Nalezené výčty jsou obvykle uvozené nadpisem, nebo větou, čeho se týkají. Takové uvození detekuje toto rozšíření, které v několika řádcích před nalezeným výčtem hledá specifické znaky (například dvojtečku) a výčet tak rozšiřuje o položky, které se vyskytují mezi nalezenou dvojtečkou a počátkem detekovaného výčtu.

4. Položka za speciálním znakem

Předmět zakázky se může skládat pouze z jedné položky. V takových případech bývá položka vyjmenovaná v řádku za nějakým specifickým znakem (například dvojtečkou). Takovou situaci detekují a v případě nálezu obsah za znakem vrací jako samostatnou položku předmětu.

5. Položka za klíčovým slovem

Obdobně jako předešlý extraktor funguje i tento s tím rozdílem, že místo specifického znaku detekuje specifické klíčové slovo (vzor regulárního výrazu).

6. Název zakázky v uvozovkách

Extraktor extrahuje obsah závorek jako název zakázky, pokud se před závorkou vyskytuje jedno z nastavených klíčových slov (např. „název“) a zároveň se nevyskytuje žádné ze zakázaných.

7. Strukturovaný název zakázky

Obdobně jako u extraktoru položky za speciálním znakem extrahuji i název zakázky. Navíc zde opět kontroluji pozitivní a negativní klíčová slova.

4.3.5.1 Výčet položek podle struktury odřádkování

Při extrakci textu se výčty položek obvykle převedou na jednotlivé položky pravidelně prokládané odřádkováním. Navíc, díky filtrování a transformaci kontextu, je taková pravidelnost až na výjimky zaručená. Na základě toho implementuji následující algoritmus, který detekuje delší úseky pravidelného vzoru odřádkování a extrahuje z něj položky:

1. Na základě délky řádků počítám strukturu odřádkování celého kontextu ve formě příznaků (skóre):
 - 0 – krátký řádek (pravděpodobně pouze odřádkování),
 - 1 – řádek potenciálně obsahující položku výčtu (delší než 5 znaků),
 - 2 – dlouhý řádek (delší než přednastavená hodnota – defaultně 100 znaků),
 - 5 – příliš dlouhý řádek (delší než 1.5 dlouhého řádku),
 - 5 – řádek přerušující sekvenci formátovací odlišností (například řádek psaný velkým písmem).
2. Zahazuji ojedinělé výskyty dlouhých řádků v souvislém bloku kratších (celkové skóre okolí nepřesahuje určitou hranici), abych nepřerušil sekvenci výčtu.
3. Z vytvořené sekvence příznaků tvořím řetězec, ve kterém hledám nejdélší posloupnost střídání 0 a 1.
4. Na základě vybrané posloupnosti extrahuji z kontextu řádky odpovídající příznakům 1.
5. Pokud jsou vybrané řádky alespoň 3, považuji takovou část za validní výčet položek a stává se výstupem algoritmu. V opačném případě extraktor nevrací nic.

4.3.5.2 Výčet položek podle odrážek

V dokumentacích se může vyskytovat více formátů výčtů položek. Klasicky se používají speciální znaky jako pomlčky a tečky, avšak existují i formáty, kde se speciálních znaků nevyužívá a seznam je členěn například jen alfanumerickými identifikátory.

Tento extraktor má za cíl podchytit všechny takové případy. Implementuju proto následující algoritmus, který detektuje znaky potenciálně tvořící výčtovou strukturu a na jejich základě extrahuje výčet položek:

1. Nalezení nejčastěji se vyskytujících počátečních dvojic znaků (nepočítaje bílé znaky). Různých dvojic označujících výčet může být více, vybírám proto dvojice, které mají alespoň minimální počet výskytů daný nastavením (defaultně 4 výskyty). Vzhledem k doméně dat, některé dvojice znaků se mohou obecně často vyskytovat na začátku vět, proto specifikují seznam zakázaných dvojic.
2. Pro každou vybranou dvojici extrahuji výčet obdobně jako u extraktoru výčtu položek podle struktury odřádkování (viz. kapitola 4.3.5.1) s modifikací:

4. EXTRAKCE VLASTNOSTÍ ZAKÁZEK

Extraktor	Pouze jeden		Bez jednoho	
	Skóre	Čas [s]	Skóre	Čas [s]
Základ/Vše	15.69	43.6	78.02	42.6
1 Výčet položek podle struktury odřádkování	45.61	40.7	68.31	44.1
2 Výčet položek podle odrážek	50.43	42.2	66.64	47.5
3 Rozšíření výčtu položek	-	-	75.13	46.9
4 Položka za speciálním znakem	38.34	43.4	77.43	45.4
5 Položka za klíčovým slovem	38.09	42.1	77.84	46.2
6 Název zakázky v uvozovkách	49.21	41.6	67.08	46.8
7 Strukturovaný název zakázky	44.15	43.1	72.63	42.4

Tabulka 4.12: Měření dopadu jednotlivých extraktorů (podle lokálních charakteristik) na dopad extrakce předmětných položek

- a) Pro výpočet struktury kontextu používám příznaky 1 pro řádky, které začínají danou dvojicí znaků a 0 jinde.
- b) Ve struktuře kontextu hledám nejdéleší posloupnost příznaků 1.
- c) Pokud jsou vybrané řádky alespoň 3, považuji takovou část za valdní výčet položek a vracím ji na výstupu.

4.3.5.3 Meření dopadu jednotlivých extraktorů

Obdobně jako dopad filtrů a transformerů v kapitole 4.3.4.3 měřím také dopad jednotlivých extraktorů na výsledek celého procesu extrakce předmětu. Měření zaznamenávám v tabulce 4.12.

V tabulce je opět vidět, že všechny extraktory mají jistý vliv na přesnost výsledku. Rozšíření výčtu položek (řádek 3) samo o sobě nelze provést, proto jsou u něj hodnoty proškrtnuté. Čas u těchto měření je bohužel zavádějící, protože výpočetní stroj byl v době měření více vytížený.

4.3.6 Extrakce podle větného rozboru

Předešlý proces – extrakce podle lokálních charakteristik – řešil extrakci do jisté míry strukturovaného textu. V kontextu předmětu tak dále zbývá ke zpracování nestrukturovaný volný text. Tento proces má za úkol z těchto zbývajících částí vyextrahovat předmětné informace, které obsahuje.

Vzhledem ke specifické povaze dat, jako jsou veřejné zakázky, se ve volném textu kapitoly předmětu zakázky vyskytují určité vzory, kterých se tímto procesem snažím využít.

Pro strojové zpracování volného textu existují nástroje řešící tokenizaci, lemmatizaci, tagování na různých úrovních, či detekci gramatických závislostí. Tyto nástroje ke svým specifickým účelům obvykle využívají naučený jazykový model.

Pro účely mé práce dělám průzkum takových technologií a modelů a pojednávám o nich v následujících kapitolách.

4.3.6.1 Jazykový model

Prague Dependency Treebank (dále jen PDT) je morfologicky, syntakticky a sémanticky anotovaný korpus českých textů[56]. PDT obsahuje anotace na třech úrovních:

1. morfologická úroveň – formy slov, tagy, lemmata,
2. analytická úroveň – povrchní syntaxe, závislosti mezi slovy,
3. tektogramatická úroveň – abstraktní lingvistický význam.

Od verze 2.0 obsahuje veliké množství českých textů se složitou a vzájemně propojenou morfologickou (2 milionů slov), syntaktickou (1,5 milionů slov) a sémantickou (0,8 milionů slov) anotací. Na sémantické úrovni navíc obsahuje anotace určitých vlastností větných struktur, víceslovných výrazů a dalších vztahů.

Universal Dependencies (dále jen UD) je framework pro sjednocené gramatické anotování textu napříč různými jazyky[8]. Z celého světa se na jeho tvorbě podílí přes 200 přispěvatelů a jejich produktem je více než 100 tzv. treebank²⁷ v až 70 různých jazycích. Český jazyk je zastoupen hned pěti treebankami s celkovými více než dvěma miliony tokenů, díky čemu dosahuje druhého (po němčině) nejobsáhlnejšího korpusu vůbec. Nejobsáhlnejší z nich – *UD_Czech-PDT* – je založen na PDT 3.0 a obsahuje přes 87 tisíc anotovaných vět (1.5 mil. tokenů).

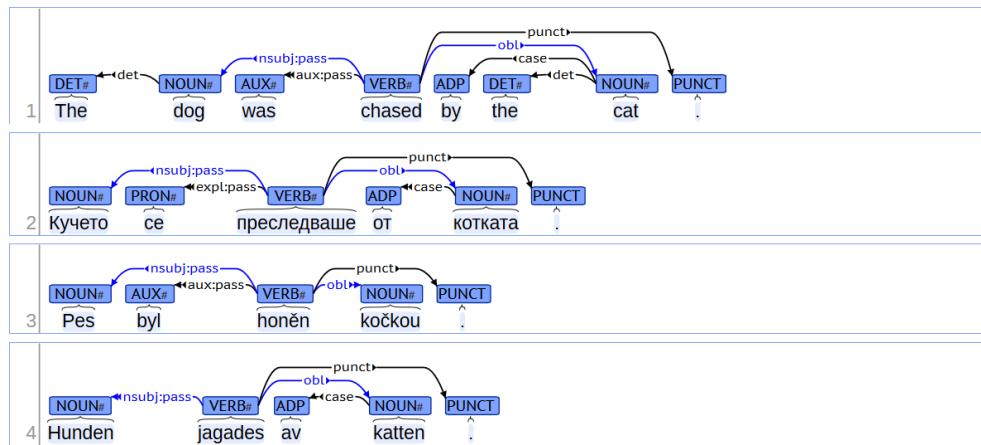
Na obrázku 4.30 je ukázka anotačního schéma UD pro několik různých jazyků.

CoNLL-U (nebo jednoduše Conllu) je textový formát pro zápis UD anotací[8]. Anotace jsou zakódované v prostém textu, čitelném jak strojově, tak i pro člověka. Anotace každého slova/tokenu je na samostatném řádku. Řádky Conllu formátu jsou tří typů:

1. řádek obsahující anotaci tokenu, která se skládá z 10 polí oddělených tabulátory,
2. prázdný řádek, označující hranice vět,
3. komentář, začínající znakem #.

²⁷Treebank – (z *CzechEncy*) Korpus syntakticky anotovaných struktur; obsahuje syntakticky anotované struktury vět v podobě závislostních stromů

4. EXTRAKCE VLASTNOSTÍ ZAKÁZEK



Obrázek 4.30: Ilustrace schéma UD anotace[8]

Věty tvoří alespoň jedna řádka s tokenem, který obsahuje následující pole:

1. ID: index ve větě,
2. FORM: původní forma,
3. LEMMA: lemma či stem,
4. UPOS: univerzální „part-of-speech“ (dále POS) tag,
5. XPOS: jazykově specifické POS tagy,
6. FEATS: výčet morfologických vlastností,
7. HEAD: předek v závislostním stromu,
8. DEPREL: závislostní vazba k předkovi,
9. DEPS: rozšířený závislostní graf,
10. MISC: další anotace.

Pro ukázku je na obrázku 4.31 vidět anotace vzorové věty.

4.3.6.2 Nástroje pro práci s jazykem

Treex je modulární NLP framework zaměřený na strojové překlady[57]. Pomocí specifických modulů lze složit scénáře, například pro tektogramatickou analýzu na úrovni PDT. Je implementovaný v jazyku Perl a využít lze ve formě webového rozhraní. Současně je již nástroj zastaralý a nahrazují ho následující.

```

1 # newdoc
2 # newpar
3 # sent_id = 1
4 # text = Pes byl honěn kočkou.
5 1 Pes pes NOUN NNMS1----A---
  Animacy=Anim|Case=Nom|Gender=Masc|Number=Sing|Polarity=Pos 3 nsubj:pass _ _
6 2 byl být AUX VpYS---XR-AA---
  Gender=Masc|Number=Sing|Polarity=Pos|Tense=Past|VerbForm=Part|Voice=Act 3 aux:pass _ _
7 3 honěn honěný ADJ VsYS---XX-AP---
  Gender=Masc|Number=Sing|Polarity=Pos|Variant=Short|VerbForm=Part|Voice=Pass 0 root _ _
8 4 kočkou kočka NOUN NNFS7----A--- Case=Ins|Gender=Fem|Number=Sing|Polarity=Pos _ _
 3 obl:agent SpaceAfter=No
9 5 . . PUNCT Z:----- _ 3 punct _ SpaceAfter=No

```

Obrázek 4.31: Ukázka CoNLL-U formátu

UDPipe je nástroj provádějící segmentizaci, tokenizaci, tagování, lematizaci a analýzu závislostí textu[58]. Pomocí nástroje lze trénovat vlastní, nebo přímo využívat předtrénované UD modely. Standardně pracuje s CoNLL-U formátem. Je optimalizovaný pro rychlosť. Je multiplatformní, podporuje rozhraní pro většinu populárních jazyků a lze ho využít i v podobě webové aplikace.

Udapi je rozraní a framework pro zpracovávání UD[59]. Je dostupný jako nástroj příkazové řádky, nebo knihovna pro jazyky Python, Perl a Java. Vnitřní reprezentace Conllu dat v Udapi je čtyřvrstvá hierarchická struktura skládající se z:

1. dokumentů – jednotlivé Conllu dokumenty,
2. svazků – (z angl. bundles) dokument může mít několik svazků, reprezentujících jednu větu dokumentu,
3. stromů – (z angl. trees) svazek může mít několik stromů, kde každý strom reprezentuje jazykovou mutaci dané věty odvozenou dle treebanky,
4. uzelů – (z angl. nodes) stromy se klasicky skládají z uzelů reprezentujících jednotlivá slova (tokeny).

conllu je odlehčená knihovna pro jazyk Python umožňující základní operace s CoNLL-U daty.

4.3.6.3 Výběr technologií a implementace

Aktuální technologický trend vývoje NLP aplikací je u UD. Pro účely extrakce předmětných položek podle větného rozkladu vybírám model *czech-pdt-ud-2.5-191206*[60], který používám v nástroji *UDPipe* pro anotaci textu a takto anotovaný text následně procesuji za použití *Udapi*.

4. EXTRAKCE VLASTNOSTÍ ZAKÁZEK

Proces extrakce podle větného rozkladu tak spočívá ve třech krocích:

1. Anotace textu a převedení do vnitřní reprezentace

Nejprve provádím anotaci pomocí *UDPipe*, který souvislý text kontextu převede do *Conllu* formátu (segmentace vět, tokenizace slov...). Následně tato data načítám do *Udapi*, čímž získávám jejich vnitřní reprezentaci.

2. Filtrování vnitřní reprezentace

Filtrování provádím větu po větě, respektive strom po stromu (vzhledem k použití pouze jedné treebanky má svazek vždy jen jeden strom). Nejprve filtroji jednoduše na základě klíčových slov týkajících se ne-relevantních informací jako jsou ceny nebo přílohy dokumentů tak, že odmažu celý podstrom těchto slov.

Následuje hlavní krok filtrování, který na základě komplexních podmínek vybírá předmětné části. Více do detailu popisují tento krok v samotné kapitole 4.3.6.4

3. Převedení zpět do textové podoby

Nakonec převádím zbývající data z *Udapi* reprezentace zpět do prostého textu.

4.3.6.4 Filtrování (ne)předmětných částí vět

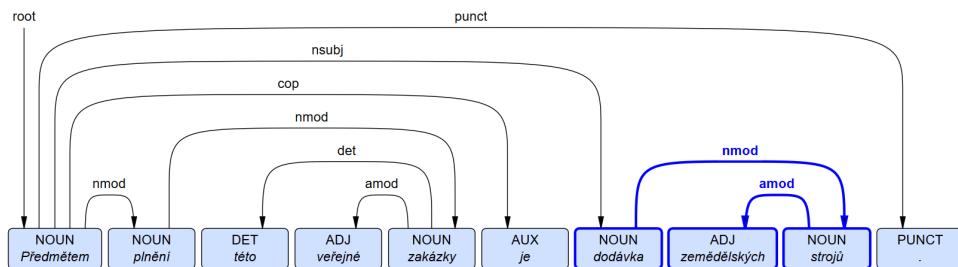
Pro úvod nejdříve uvedu příklad, na kterém nastíním myšlenku a proces tohoto filtru.

Mějme větu: "Předmětem plnění této veřejné zakázky je dodávka zemědělských strojů.", jejíž rozklad je na obrázku 4.32. Předmětná část této věty je „dodávka zemědělských strojů“. Otázkou je, jakým vzorem takovou část podchytit?

Závislostní vazba uzlu „dodávka“ na předchůdce („předmětem“) je *nsubj*. Zdá se, že by se v tom případě dalo odchytávat koncové uzly vazby *nsubj* a jejich podstromy.

Bohužel, ne vždy je situace takto přímočará. Treebanky jsou navíc omezeně rozsáhlé a dochází k chybám anotace, které jsou někdy kritické.

Obecně, algoritmus výběru části vět zakládám na selekcii kandidátních uzel a následném doplnění o jejich kontext.

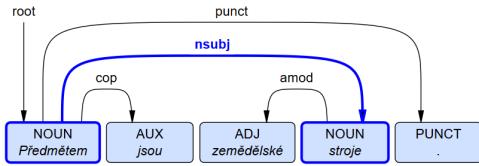


Obrázek 4.32: Ukázka rozkladu předmětné věty (předmětná část věty je zvýrazněna modře)

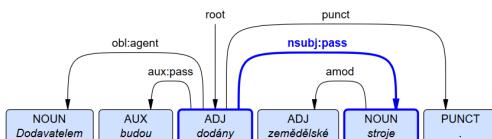
Selekce kandidátních uzlů je řízena následujícími podmínkami:

- vyhovující závislostní vazby (větné členy) omezuji na²⁸:
 - *nsubj* (podmět; z angl. „nominal subject“)
např. „Předmětem jsou zemědělské **stroje**.“ (viz. obrázek 4.33),
 - *nsubj:pass* (pasivní podmět; z angl. „passive nominal subject“)
např. „Dodavatelem budou dodány zemědělské **stroje**.“ (viz. obrázek 4.34),
 - *obj* (předmět; z angl. „direct object“)
např. „Dodavatel dodá zemědělské **stroje**.“ (viz. obrázek 4.35),
 - *obl* (doplňek; z angl. „oblique argument or adjunct“)
např. „Dodavatel se zavazuje k **dodání** zemědělských strojů.“ (viz. obrázek 4.36),
 - *obl:arg* (předmět; z angl. „oblique argument“)
např. „Předmět spočívá v **dodání** zemědělských strojů.“ (viz. obrázek 4.37),
 - *nmod* (přívlastek neshodný; z angl. „nominal modifier“)
např. „Předmětem je **dodání** zemědělských strojů.“ (viz. obrázek 4.38).
- specifikují množinu zakázaných slov v podobě lemat (např. *smlouva*, *předmět*, *specifikace*, *dokumentace*, *uzavření*, *dodatek*, *podmínka*...),
- podobně specifikují množinu zakázaných slov, které nesmějí předcházet (např. *dodatečný*, *nutný*...),
- nakonec specifikují slovesa, která by měla předcházet (např. *zavazovat*, *dodat*, *zajistit*, *provést*...).

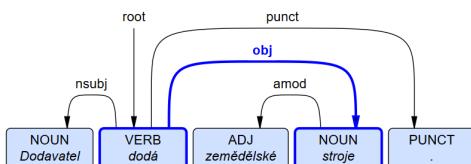
4. EXTRAKCE VLASTNOSTÍ ZAKÁZEK



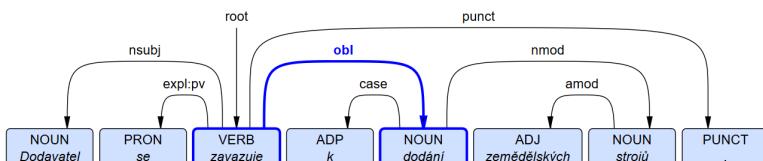
Obrázek 4.33: Ukázka předmětné části jako vazby *nsubj*



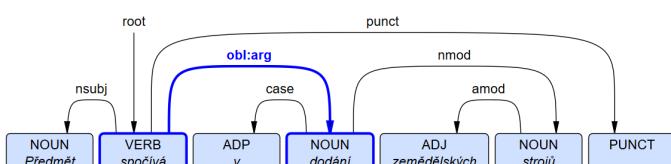
Obrázek 4.34: Ukázka předmětné části jako vazby *nsubj:pass*



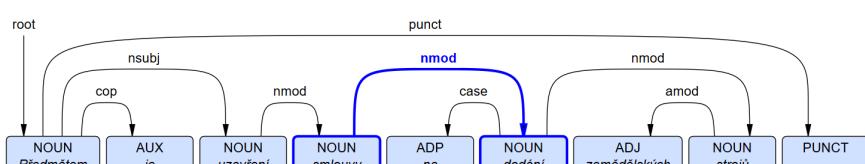
Obrázek 4.35: Ukázka předmětné části jako vazby *obj*



Obrázek 4.36: Ukázka předmětné části jako vazby *obl*



Obrázek 4.37: Ukázka předmětné části jako vazby *obl:arg*



Obrázek 4.38: Ukázka předmětné části jako vazby *nmod*

Pokud je ve větě více cílových kandidátních slov v konjunktivním vztahu (*conj*), předchozí selekce vybere často pouze první z nich. Proto každé kandidátní slovo rozšířuji o jeho konjunktivní doplněk.

Takto vybraná kandidátní slova rozšířuji o jejich kontext pro výběr celé části věty. Kontext obvykle vybírám jako podstom daného kandidátního slova. Celé vybrané části ještě dodatečně filtruji vzhledem k různým vlastnostem, aby například netvořily příliš dlouhé řetězce. Jako vyfiltrovaný výstup pro danou větu spojuji všechny takto vybrané části.

4.3.7 Sestavení komponenty pro extrakci předmětu

Komponentu pro extrakci předmětu sestavuji podle návrhu celého procesu (viz. 4.3.1) tak, že provádí kroky:

1. Identifikace a extrakce kontextů předmětu (4.3.3)
2. Předzpracování jednotlivých kontextů na základě lokálních charakteristik (4.3.4)
3. Extrakce položek z předzpracovaného kontextu (4.3.5)
4. Anotace vyfiltrovaného kontextu (4.3.6.3)
5. Filtrování anotovaného kontextu podle větného rozkladu (4.3.6.4)
6. Extrakce zbylých položek vyfiltrovaného anotovaného kontextu
7. Sloučení obou skupin položek

Komponenta přijímá jakýkoli souvislý text a vrací formátovaný text obsahující odřádkované jednotlivé položky.

²⁸Jednotlivé definice vazeb jsou dostupné v angličtině v dokumentaci UD[8].

4.4 Extrakce lokality a předmětu podnikání zadavatelů

V kapitolách „Data o lokacích“ (3.4) a „Data o předmětu podnikání zadavatelů“ (3.5) uvádím možnost získání informací o lokalitě a předmětu podnikání zadavatelů zakázek.

Na základě toho implementuji komponentu, která pro jednotlivé zadavatele posílá dotaz podle čísla IČO na rozhraní systému ARES pro získání informací o daném zadavateli. Z odpovědi komponenta extrahuje oboje adresu i předmět podnikání.

Pro další zpracování adresy implementuji geokodér spočívající v dota-zování se na službu geokódování portálu Mapy.cz.

Získaný předmět podnikání rozdělují podle odřádkování do jednotlivých položek.

Metrika podobnosti zakázek

Jednou z nezbytných částí doporučovacích systémů je metrika podobnosti položek, které systém doporučuje. Správně zvolená/navržená metrika musí reprezentovat důležité vlastnosti porovnávaných položek, avšak tyto vlastnosti se mohou lišit v závislosti na úloze. Žádná metrika není univerzálně použitelná ve všech případech.

V doméně veřejných zakázek se můžeme ptát na otázku: jaké zakázky jsou si podobné? Pro účely mé práce definuji následující vlastnosti pro určení takové podobnosti: předmět plnění zakázky, lokalita zadavatele a předmět podnikání zadavatele.

Zatímco podobnost lokality zadavatele lze intuitivně řešit přímo jako invertzní funkce geografické vzdálenosti dvou bodů (např. podle sídla zadavatele), předmět podnikání zadavatele a samotný předmět plnění zakázky již takto přímo měřit nelze. Podobnost předmětu zakázek by mohla jít řešit na základě CPV kódů, avšak jen do omezeného rozsahu jejich hierarchické struktury. Pro obecné řešení se tak nabízí řešení podobnosti předmětu specifikovaného v textu dokumentace zakázek.

5.1 Metrika podobnosti textu

Existuje mnoho algoritmů pro porovnávání textových řetězců. Podle základní myšlenky se člení na algoritmy založené na[61]:

1. editační vzdálenosti

Metoda spočívá v počítání operací nutných pro transformaci jednoho řetězce na druhý. Čím více je potřeba operací, tím méně jsou si řetězce podobné.

Příkladem jsou algoritmy Hammingovy či Levenshteinovy vzdálenosti.

2. množinových operacích

Algoritmy této kategorie dělí řetězce na množiny tokenů a hledají se podobné tokeny z obou množin. Čím více podobných tokenů množiny mají, tím jsou si samotné řetězce více podobné.

Zástupcem této metody je tzv. „Jaccard index“, který používám při validacích algoritmů pracujících s textem. Tento index udává podobnost dvou množin jako podíl jejich průniku se součinem, viz. předpis 5.1.

$$J(A, B) := \frac{|A \cap B|}{|A \cup B|} \quad (5.1)$$

3. hledání společných sekvencí

U této kategorie spočívá metrika v hledání společných sekvencí obou řetězců.

Například algoritmus Ratcliff-Obershelpovy vzdálenosti využívá binární dělení a hledání nejdelších podřetězců v obou větvích.

Všechny tyto algoritmy jsou zaměřené pouze na podobnosti řetězců samotných, což je postačující v případech, kdy jsou si tyto řetězce poměrně podobné. To nastává například při validaci výsledků transformace textu oproti referenční hodnotě.

V případě, kdy je potřeba řešit významová (sémantická) podobnost textů však tyto metody selhávají. Sémantická podobnost textů je netriviální problém sám o sobě, dnes běžně řešený za pomocí modelů strojového učení. U řešení sémantické podobnosti se hlavní problém přenáší na transformaci obsahu textu do prostoru vlastností (vektorového prostoru), ve kterém lze vzdálenost (podobnost, souvislost) měřit. Příkladem takové transformace je proces tzv. „embeddingu“ dokumentů, který detailně popisuje v samotné kapitole 4.2.

Pro měření vzdálenosti dokumentů převedených do vektorového prostoru je vhodná kosinová vzdálenost, jak popisuje do detailu ve svém článku[62] Emery. Oproti euklidovské vzdálenosti kosinová vzdálenost potlačuje odchylky způsobené absolutní délkou textů.

5.2 Komponenty pro vyhodnocování podobnosti

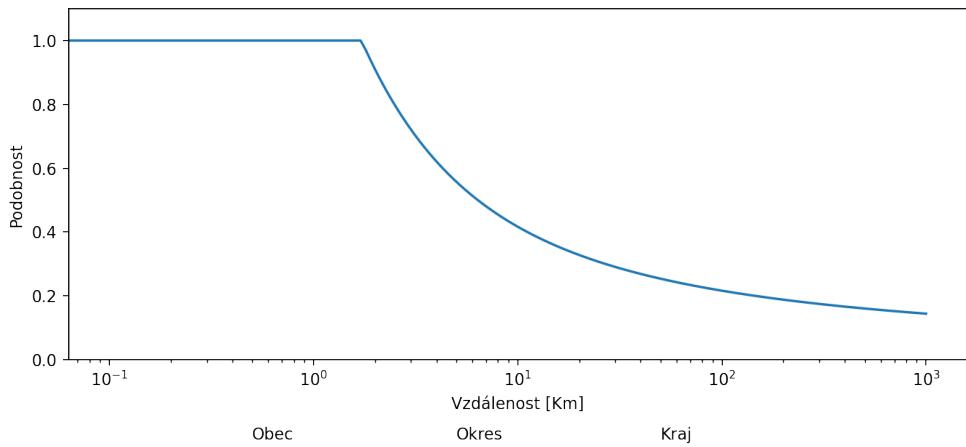
Pro použití v systému implementuji následující komponenty.

5.2.1 Výpočet Jaccard indexu

Komponenta využívá knihovny *textdistance*²⁹, která poskytuje všechny algoritmy zmíněné v kapitole 5.1 a mnoho dalších. Knihovní funkce přijímá

²⁹textdistance - Python knihovna; dostupná z <https://pypi.org/project/textdistance/>

5.2. Komponenty pro vyhodnocování podobnosti



Obrázek 5.1: Ukázka průběhu geografické podobnosti 5.2

dvě kolekce tokenů a vrací jejich podobnostní index. Tokenizaci obou řetězců provádím jednoduše rozdělováním podle mezer. Knihovna pracuje s množinami jako s tzv. multimnožinami, tedy bere v úvahu počty výskytů.

5.2.2 Výpočet geografické podobnosti

Komponenta počítá vzdálenost mezi cílovým a referenčním GPS bodem a následně odvozuje jejich „podobnost“. Pro výpočet vzdálenosti (geodezické³⁰) využívám knihovny *geopy*³¹, která – mimo jiné – poskytuje různé algoritmy počítání geografické vzdálenosti. Knihovní funkce přijímá dva GPS body a vrací metrickou vzdálenost. Vzdálenost v kilometrech převádí na podobnost standardizací danou předpisem 5.2, kde d je vypočítaná nenulová vzdálenost a $\log1p$ je speciální logaritmická funkce knihovny *numpy*, definovaná jako přirozený logaritmus z $x+1$.

$$Sim(d) := \min\left(1, \frac{1}{\log1p(d)}\right) \quad (5.2)$$

Tato standardizace má za důsledek, že vzdálenosti do 1,7 kilometru dávají podobnost 1, což zhruba odpovídá logické podobnosti na úrovni stejné obce, jako je znázorněno v diagramu 5.1.

Komponenta přijímá na vstupu tzv. „*pandas*³² dataframe“ pro cílovou a referenční množinu bodů a na výstupu vrací mapu n nejbližších referenčních bodů pro každý z cílových.

³⁰Geodezická vzdálenost - vzdálenost po povrchu Země jako elipsoidu

³¹geopy - Python knihovna; dostupná z <https://pypi.org/project/geopy/>

³²pandas - Python knihovna; dostupná z <https://pypi.org/project/pandas/>

5.2.3 Výpočet kosinové podobnosti

Komponenta, obdobně jako předchozí, počítá vzdálenost mezi cílovým a referenčním bodem (vektorem) a následně odvozuje jejich „podobnost“. Pro výpočet (kosinové) vzdálenosti vektorů podle definice 5.3 implementuji komponentu založenou na maticových operacích pomocí nástrojů knihovny *numpy*. Komponenta přijímá dvě kolekce vektorů a vrací kosinovou vzdálenost všech párů vektorů mezi těmito dvěma kolekcemi. Komponenta vrací vzdálenost v intervalu [0,2], kterou převádí na podobnost standardizací danou předpisem 5.4, kde d je vypočítaná vzdálenost.

$$CosDist(\mathbf{A}, \mathbf{B}) := 1 - \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\| * \|\mathbf{B}\|} \quad (5.3)$$

$$Sim(d) := \frac{2 - d}{2} \quad (5.4)$$

Pro výpočet kosinové vzdálenosti nepoužívám knihovní funkce z důvodu optimalizace výpočtu pomocí částečného předpočítání matic působících ve výpočtu. Výsledek optimalizace je zaznamenaný v tabulce 5.1. Měření je prováděno na kolekcích 100 cílových vektorů (měnících se) oproti 10 000 referenčních vektorů (stálých) o dimenzi 300. Výpočty jsou prováděny ve 100 epochách pro podchycení amortizační optimalizace předpočítání stálých matic.

Knihovna	Funkce	Čas [s]
Sklearn ³³	<i>cosine_similarity</i>	5.86
SciPy ³⁴	<i>cdist/cosine</i>	44.50
—	vlastní	4.26

Tabulka 5.1: Měření doby výpočtu algoritmů pro výpočet kosinové vzdálenosti

³³Sklearn (scikit-learn) – Python modul pro strojové učení; dostupný z <https://pypi.org/project/scikit-learn/>

³⁴SciPy – Python knihovna pro matematické, vědecké a inženýrské úlohy; dostupná z <https://pypi.org/project/scipy/>

5.2.4 Výpočet „euklidovské“ podobnosti

Přestože pojem „euklidovské podobnosti“ není přesně definován, euklidovská vzdálenost (jinak zvaná jako euklidovská metrika) je obecně známý pojem a lze ji různými způsoby na podobnost převést.

Oproti předešlé metodě tato komponenta počítá (euklidovskou) vzdálenost vektorů pomocí funkce *euclidean_distances* knihovny *Sklearn*. Knihovní funkce přijímá dvě kolekce vektorů a vrací euklidovskou vzdálenost všech párů vektorů mezi těmito dvěma kolekcemi. Euklidovská vzdálenost nabírá oboru hodnot kladných reálných čísel a proto ji na podobnost převádí jednoduchou inverzní transformací podle předpisu 5.5.

$$Sim(d) := \frac{1}{d} \quad (5.5)$$

Podobně jako pro kosinovou vzdálenost se pokouším implementovat optimalizaci výpočtu euklidovské vzdálenosti na základě vztahu 5.6[63]. Výsledek je ovšem neúspěšný, jak je zaznamenáno v tabulce 5.2.

$$EucDist(\mathbf{A}, \mathbf{B}) := \|\mathbf{A} - \mathbf{B}\|^2 = -2\mathbf{ab} + \mathbf{aa}^T + \mathbf{bb}^T \quad (5.6)$$

Knihovna	Funkce	Čas [s]
<i>Sklearn</i>	<i>euclidean_distances</i>	4.06
<i>SciPy</i>	<i>cdist[euclidean]</i>	44.70
—	vlastní	4.75

Tabulka 5.2: Měření doby výpočtu algoritmů pro výpočet euklidovské vzdálenosti

KAPITOLA **6**

Doporučovací systém

V dnešní době je na internetu nepřeberné množství informací, zdrojů, věcí a mnoho dalšího. K efektivnímu využití takového obsahu je zapotřebí systémů, které uživateli budou schopny data třídit do rozumně rozsáhlé množiny relevantních položek. Takové systémy se nazývají „doporučovací“ a vyskytují se v podstatě ve všech populárních on-line platformách jako jsou Youtube, Amazon, Netflix, Facebook a mnoho dalších.

Jak shrnuje ve svém článku[64] autor, ke stavbě doporučovacích systémů se používají dvě základní paradigmata: tzv. „kolaborativní filtrování“ (dále angl. collaborative filtering) a „obsahově založené“ metody (dále angl. content-based).

6.1 Collaborative filtering

Metody kolaborativního filtrování jsou založené výhradně na předešlých interakcích uživatele s položkami systému. Interakce systém zaznamenává do tzv. „uživatel–položka matice“ (z angl. user–item matrix), na základě které systém zároveň detekuje podobné uživatele či položky.

Existují dvě základní kategorie algoritmů:

1. Memory based

Tzv. „memory based“ algoritmy jsou založené na přímém využití matice pro výpočet nejpodobnějších záznamů.

2. Model based

Metody tzv. „model based“ přístupu oproti tomu využívají odvozeného modelu, který zobecňuje zaznamenané interakce a snaží se v nich nalézt relevantní informaci pro provádění predikce.

Výhoda kolaborativního filtrování je, že neklade žádné nároky na vstupní informace o uživatelích nebo položkách. Zároveň jsou nativně schopny efekti-

6. DOPORUČOVACÍ SYSTÉM

vitu doporučování zlepšovat v průběhu fungování s přibývajícím počtem zaznamenaných interakcí.

Tato vlastnost ovšem přináší i jejich hlavní problém tzv. „chladného startu“ (z angl. cold start problem), který představuje situaci, že novému uživateli nelze nic doporučovat, stejně jako nelze doporučovat právě přidané položky. Existují různá řešení tohoto problému, mezi která patří například i použití nekolaborativní metody.

6.2 Content based

Jak napovídá název, metody tohoto typu jsou založené na obsahu, respektive vlastností obou uživatelů i položek, na základě kterých se staví model. Podle toho, které vlastnosti se využívají dělíme dva přístupy:

1. Item-centered

Tzv. „item-centered“ přístup využívá uživatelských vlastností pro predikci relevance dané položky ke všem uživatelům.

2. User-centered

„User-centered“ přístup naopak používá vlastnosti položek pro predikci jejich relevance pro daného uživatele.

Metody obou přístupů lze zkombinovat tak, že se pro sestavení modelu využívají vlastnosti obou entit.

6.3 Vyhodnocování doporučovacích systémů

Vyhodnocování doporučovacích systémů je poněkud složité, přestože lze do určité míry použít standardní metody vyhodnocování algoritmů strojového učení. V případě, že je doporučovací model učený na množině zaznamenaných interakcí, lze použít klasických metrik chybovosti či přesnosti pro jeho automatické vyhodnocování.

U doporučování hraje důležitou roli vyrovnaný poměr explorace a exploitation. Kromě pouhého získání nejbližšího výsledku nás tak zajímá i jeho různorodost. Na důvěryhodnost systému má také dopad vysvětlitelnost jednotlivých doporučení. Takové, a jiné vlastnosti je možné vyhodnocovat za běhu systému při zaznamenávání reálného chování uživatelů na jednotlivé doporučené položky.

6.4 Návrh doporučovacího algoritmu

V kapitole často používám pojmenování „cílová“ či „referenční“ matici, vektor, položky apod. Pojemem „cílová“ označuji entitu, na základě které systém dohledává entity „referenční“. Například tak může vzniknout cílový dotaz obsahující cílovou položku „zemědělské stroje“, která se převádí do reprezentace cílového vektoru či matice. Dotaz se dotazuje oproti referenčním položkám v referenčním datasetu (databázi), které se taktéž převádí do reprezentace referenčních vektorů respektive matice.

Vzhledem ke zmíněnému problému chladného startu kolaborativních metod, který by v doméně veřejných zakázek měl kritický dopad volím pro doporučovací algoritmus metodu založenou na obsahu (viz. kapitola 6.2).

Povaha uživatelů (zadavatelů) a položek (zakázek) je do značné míry související ve smyslu podobnosti předmětu zakázek s předmětem podnikání zadavatelů či místnímu zařazení obou entit. Návrh algoritmu tak zakládám na všech těchto podobnostech zkombinováním vlastností obou uživatelů i položek.

Algoritmus je navržený pro zpracování obecného cílového dotazu na referenční dataset. Dotaz může být zadán jak ve smyslu konkrétního vyhledávání, tak ve smyslu doporučování na základě uživatelského (preferenčního) profilu. Skládání obou cílového dotazu a referenčního datasetu na základě patřičných vlastností je zobrazeno na schéma 6.1. Vlastnosti jsou reprezentované jednotlivými barvami.

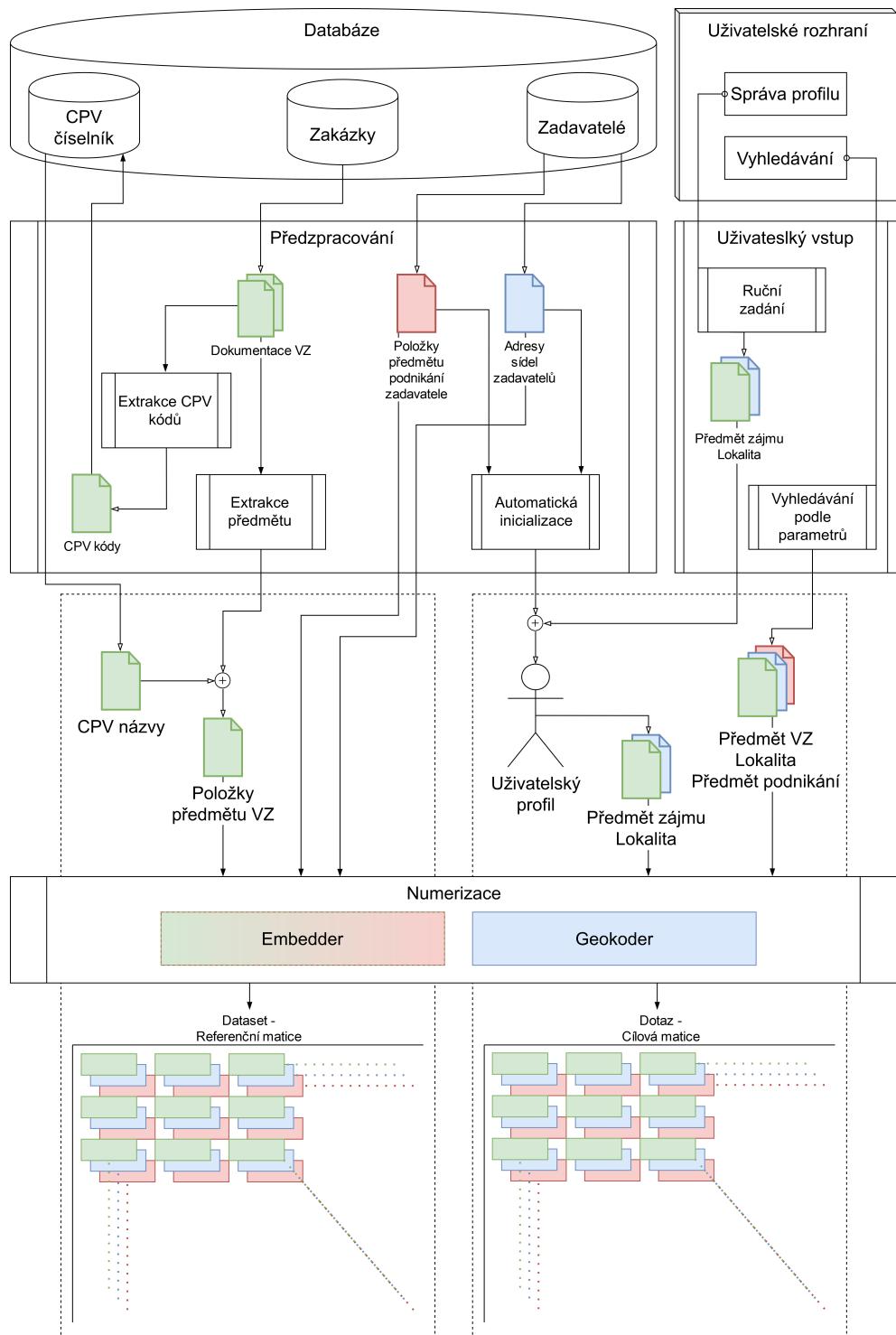
Uživatelský profil se skládá ze zájmových položek a lokality uživatele. Profil může být inicializován na základě údajů zadavatele identifikovaného podle čísla IČO, nebo zadán ručně uživatelem, jak je zobrazeno na schéma 6.1.

Před samotným započetím operace výpočtu podobnosti jsou všechny vlastnosti numerizovány.

Celý algoritmus doporučování se skládá z jednotlivých komponent, kde každá se zaměřuje na určitou vlastnost. Jednotlivé komponenty lze využít samostatně, nebo je libovolně kombinovat s různými váhami.

V následujících kapitolách popisují jednotlivé komponenty podle vlastnosti, na kterou se zaměřují.

6. DOPORUČOVACÍ SYSTÉM



Obrázek 6.1: Schéma skládání vlastností pro cílový dotaz a referenční dataset. Jednotlivé vlastnosti jsou reprezentované barvami: zelená – předmět, červená – předmět podnikání zadavatele, modrá – lokalita.

6.4.1 Předmět zakázky

Předmět zakázky je hlavní vlastnost, na základě které je celý systém postaven. V kapitole 4.3 popisuji dopodrobna způsob extrakce předmětu zakázek do podoby jednotlivých položek. Na schématech 6.1 a 6.2 je reprezentovaná zelenou barvou.

Kromě samotných položek předmětu extrahuji i CPV kódy, jak popisuji v kapitole 4.1.4. Názvy těchto kódů využívám jako dodatečné položky předmětu.

Každá položka je embeddovaná pomocí embedderu (kap. 4.2.8) do vektorové reprezentace, ve které ji dále používám ve smyslu referenčního vektoru komponentou pro výpočet vektorové podobnosti (kap. 5.2.3).

Na předměty jednotlivých zakázek v podobě referenčních vektorů sestavují dotaz ve stejné reprezentaci cílových vektorů složený z:

1. jedné či více položek přímého vyhledávání,
2. zájmových položek uživatelského profilu.

Komponenta výpočtu vektorové podobnosti vrací jako výsledek matici podobnosti cílových položek dotazu vůči referenčním položkám zakázek. Tyto podobnosti agreguji váženým průměrem podle jednotlivých zakázek, kde váhy položek jsou podobnosti samy o sobě (pro umocnění podobnějších položek oproti nepodobným). Touto agregací získávám relevanci zakázky podle jejího předmětu, jako je zobrazeno na schéma 6.2.

Implementace této komponenty je adaptována pro zpracování dotazů více uživatelů najednou. V takovém případě jsou výsledky nejdříve agregovány podle samotných uživatelských dotazů.

6.4.2 Lokalita zakázky

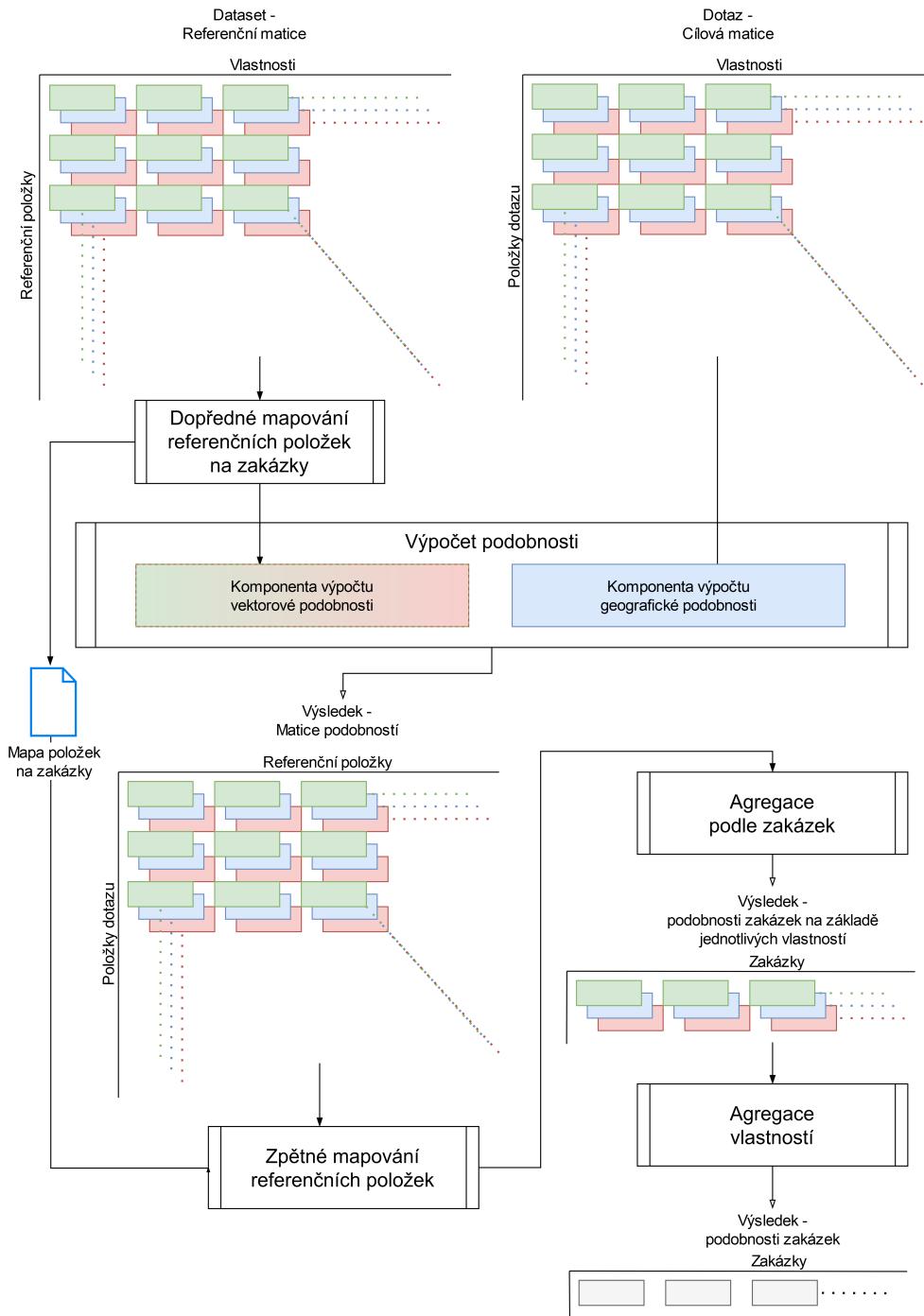
Lokalita zakázky je doplňková vlastnost rozšiřující možnosti dotazování. Vlastnost spočívá v identifikaci adresy sídla zadavatele zakázky, jak zmiňuji v kapitole 4.4. Tato vlastnost je na schématech 6.1 a 6.2 reprezentovaná modře.

Adresu sídla každého zadavatele převádí na GPS souřadnice komponentou 4.4.

Z důvodu, že se výpočet geodetické vzdálenosti ukazuje být příliš náročný (viz. tabulka 6.1) pro počítání mezi početnějším množstvím bodů a přesná vzdálenost není pro účely komponenty nezbytná, implementují approximační algoritmus výpočtu vzdálenosti v euklidovském prostoru.

Approximace spočívá v převedení GPS souřadnic na souřadnice 2D prostoru vynásobením patřičné souřadnice vzdáleností odpovídající jednomu stupni na území České republiky, které jsou 111.32 kilometru na jeden stupeň zeměpisné šířky (všude na Zemi) a 71.94 kilometru na jeden stupeň zeměpisné délky na úrovni ČR (experimentálně zjištěno optimalizací oproti geodetické vzdálenosti na množině 100 cílových a 1000 referenčních bodů náhodně vygenerovaných

6. DOPORUČOVACÍ SYSTÉM



Obrázek 6.2: Schéma výpočtu podobnosti

v okolí geografického těžiště ČR). Optimalizovaným parametrem dosahují na dané množině bodů odchylky 0.68 % za řádově tisícinásobného zrychlení (viz. výsledky tabulky 6.1).

Odvozené 2D souřadnice dále využívám komponentou výpočtu euklidovské podobnosti (viz. kapitola 5.2.4).

Knihovna	Funkce	Čas [s]
<i>geopy</i>	<i>geodesic</i>	25.0000
<i>geopy</i>	<i>great_circle</i>	1.9500
—	vlastní	0.0035

Tabulka 6.1: Měření doby výpočtu algoritmů pro výpočet geografické vzdálenosti

Na lokalitu jednotlivých zakázek v podobě referenčních bodů sestavuji dotaz ve stejné reprezentaci cílových bodů složený z:

1. adresy či GPS souřadnic přímého vyhledávání,
2. lokality danou uživatelským profilem.

Komponenta výpočtu geografické podobnosti vrací jako výsledek matici podobnosti cílového bodu vůči referenčnímu bodu každé zakázky.

Implementace této komponenty je adaptována pro zpracování dotazů více uživatelů najednou. V takovém případě jsou výsledky nejdříve agregovány podle samotných uživatelských dotazů.

6.4.3 Předmět podnikání zadavatele

Předmět podnikání zadavatele zakázky je doplňková vlastnost dále rozšiřující možnosti dotazování. Vlastnost spočívá v identifikaci předmětu podnikání zadavatele zakázky, jak zmiňuji v kapitole 4.4. Na schématech 6.1 a 6.2 je tato vlastnost reprezentovaná červeně.

Dále pracuji s touto vlastností totožně jako s předmětem zakázky, tedy – embedding – výpočet vektorové podobnosti – agregace (viz. kapitola 6.4.1).

6.4.4 Kombinace vlastností

Jak zmiňuji v úvodu kapitoly 6.4, vlastnosti lze kombinovat pro složení komplexního dotazu.

Skládání dotazu a datasetu v tom případě probíhá najednou složením daných vlastností do jedné kolekce, přičemž komponenty výpočtu podobnosti jednotlivých vlastností využívají jim odpovídající vlastnost nezávisle na ostatních. Výpočet podobnosti a agregace probíhají totožně pro každou ze složených vlastností.

6. DOPORUČOVACÍ SYSTÉM

Navíc je zde akorát poslední krok – agregace vlastností. Tento krok se provádí na úrovni jednotlivých zakázek tak, že se z výsledků podobnosti jednotlivých vlastností počítá vážený průměr.

Aplikace

Doporučovací systémy se tradičně uplatňují v prostředích s vysokým počtem přistupujících uživatelů. Příkladem jsou populární online platformy jako je Youtube, Netflix, Facebook, Spotify a jiné. Pro demonstraci systému realizovaného v mé práci tak vytvářím webovou aplikaci, která poskytuje uživatelské rozhraní k jeho použití. V této kapitole stručně shrnuji vlastnosti a možnosti tohoto rozhraní, které podchycují funkčnost celého systému.

7.1 Funkce

Aplikace disponuje dvěma základními funkcemi:

1. vyhledávání zakázek – vyhledávání konkrétních zakázek na základě parametrů zadaných uživatelem; je nezávislé na konkrétním uživateli,
2. doporučování zakázek – automatické nabízení vybraných zakázek bez nutnosti exaktního uživatelského vstupu; řídí se profilem uživatele.

Tyto dvě funkce se principiálně odlišují pouze stylem získání vstupu, na základě kterého systém dohledává relevantní zakázky. V obou případech je výsledkem množina několika zakázek, které aplikace ve stručné reprezentaci zobrazuje na stránce v podobě seznamu s možností prostopu do detailu jednotlivých zakázek. V detailu jsou poté kompletně zobrazeny všechny informace dané zakázky s odkazem na profil³⁵ zadavatele, kde je zveřejněná.

Vyhledávání je možné provádět bez nutnosti přihlášení, zatímco funkce doporučování je přihlášením podmíněna. Uživatelské přihlašování aplikace simuluje na základě unikátního identifikátoru nebo čísla IČO, podle kterého inicializuje uživatelský profil.

³⁵profil zadavatele – nezaměňovat s uživatelským profilem; více viz. kapitola 3.1

7. APLIKACE

V případě, že je uživatel přihlášen, na hlavní stránce aplikace jsou mu nabízeny zakázky ve čtyřech sekcích:

1. Zakázky z blízkého okolí – podle lokality uvedené na profilu a lokality zadavatelů zakázek (kap. 6.4.20),
2. Zakázky s předmětem zájmu – podle položek zájmu uvedených na profilu a předmětu jednotlivých zakázek (kap. 6.4.1),
3. Zakázky podobných zadavatelů – též podle položek zájmu, ale oproti předmětu podnikání zadavatelů zakázek (kap. 6.4.3),
4. všechna tři kritéria dohromady.

Aplikace umožňuje uživateli informace svého profilu upravovat a měnit tak svou lokalitu a předmět zájmu.

Podle prostupů na detail jednotlivých zakázek aplikace aktualizuje profil daného uživatele tak, aby se doporučování přizpůsobovalo jeho chování. To řešíme přidáváním položek předmětů daných zakázek do uživatelského profilu jako doplňkové položky zájmu, které jsou také vidět na profilu uživatele.

7.1.1 Technologie

Vzhledem k převážnému používání jazyku Python v ostatních částech práce tento jazyk používám i pro vytvoření webové aplikace, přičemž aplikaci stavím konkrétně pro její flexibilitu a jednoduchost na technologii *Flask*³⁶.

V rámci celé práce používám databázi *Postgres* pro průběžné ukládání zpracovávaných dat, ke které vytvářím přístupovou komponentu za použití knihovny *psycopg2*³⁷. Pro napojení aplikace na databázi tak plně využívám této komponenty.

Inicializace aplikace je závislá na konfiguraci předané speciálním souborem, ve kterém jsou specifikované parametry:

- úroveň a název souboru pro logování,
- nastavení pro inicializaci uživatelské „session“,
- přístupové údaje k databázi,
- cesta k modelu pro embedder (viz. kapitola 4.2.8).

³⁶Flask – Python webový framework;
dokumentace dostupná z <https://flask.palletsprojects.com/en/1.1.x/>

³⁷psycopg2 – Python adaptér pro Postgres databázi;
dostupný z <https://pypi.org/project/psycopg2/>

Budoucí rozšíření systému

V práci se mi podařilo zprovoznit prototyp doporučovacího systému, který do určité míry funguje. Pro nasazení v reálném prostředí by však bylo vhodné či přímo nutné systém rozšířit o další funkčnosti. O těchto rozšířených diskutuji v této kapitole v pořadí od rozšíření s nejvyšší prioritou po rozšíření méně důležitá.

8.1 Aktualizace dat

V práci jsem používal různé datasety, které byly více či méně aktuální. Ani zdaleka jsem však neobsáhl celou množinu dat českých veřejných zakázek.

Produkční provoz systému by byl závislý na kompletním datasetu, který se na denní bázi průběžně rozšiřuje. Meziročně v česku přibývá zhruba 50 tisíc zakázek, což vychází na více jak 100 zakázek denně. Za předpokladu průměru 10 dokumentů na zakázku to poté vychází zhruba na půl milionu dokumentů za rok.

V rámci práce jsem vytvořil databázi dokumentů zakázek za období leden až červen 2019. Celkem 900 tisíc dokumentů ke 130 tisícům zakázek, kde se ovšem dle mého zjištění vyskytuje značné množství duplicit. Tato databáze s dokumenty ve formátu prostého textu přesahuje velikosti 4 GB. Lze tedy předpokládat, že s přírůstkem 50 tisíc zakázek za rok bude velikost databáze úměrně růst rychlostí alespoň nízkých jednotek GB za rok.

Data je ovšem potřeba udržovat co nejaktuálnější, což by pravděpodobně shořešit každodenními aktualizacemi databáze zakázek, které by mohly být prováděny například v pravidelných nočních „jobech“.

8.2 Aplikace

Z pohledu uživatele je aplikace nejdůležitější částí celého systému. V práci jsem vytvořil jednoduchou webovou aplikaci pro demonstrační účely systému,

8. BUDOUCÍ ROZŠÍŘENÍ SYSTÉMU

avšak pro reálné použití je nedostačující.

Pro dosažení maximální uživatelské spokojenosti by bylo potřeba vytvořit webovou aplikaci za použití vhodných moderních webových a bezpečnostních technologií. Jedním ze slabých míst současného řešení je též komponenta pro komunikaci s databází, která se jeví jako tzv. úzké hrdlo v odevzvě systému.

Uživatelský profil současně aplikace pouze simuluje, což by bylo potřeba nahradit kompletním robustním řešením.

8.3 Pokročilejší algoritmy výpočtu podobnosti

Přestože je úzkým hrdlem současné implementace aplikace databázová vrstva, je možné, že by se výkonné problémy dotazování na databázi podařilo vyřešit a nejpomalejší částí systému by se tak stal samotný výpočet podobnosti.

V práci jsem řešil optimalizaci výpočtu podobnosti za použití funkcí pro maticové operace knihovny *numpy*, která se zaměřuje na efektivitu výpočtů. Touto implementací trvá výpočet podobnosti obyčejného dotazu oproti zhruba milionu položkám (50 tisíc zakázek s průměrným počtem 20 položek na zakázku, což odpovídá množství dat za jeden rok) řádově jednotky sekund.

Pokud by byla efektivita současné implementace v produkci nedostačující, bylo by možné pro snížení velikosti počítaných matic algoritmus výpočtu podobnosti rozšířit o „clusterování“ či agregaci položek. V případě clusterování by se podobnost počítala nejdříve na úrovni jednotlivých clusterů a poté k jednotlivým položkám. Agregace položek by mohla probíhat za cenu ztráty určité informace jak na úrovni uživatelského profilu, tak na úrovni zakázek.

8.4 Diverzifikace a explorace doporučování

Současný algoritmus doporučování je tzv. „hladový“, tedy zaměřuje se pouze na nalezení několika nejpodobnějších položek (příliš vysoká exploatace), což pro reálné použití není dobrá vlastnost.

Bylo by vhodné aplikovat některé metody pro balancování poměru explorace a exploatace. Nejjednodušším řešením by mohla být aplikace náhodného vlivu pro výpočet podobnosti položek, která by měla za důsledek větší diverzifikaci výsledků. Pokročilejší metodou by potom mohlo být například clusterování podobných položek za účelem vybírání pouze podmnožiny výsledků z daného clusteru.

8.5 Možnosti filtrování zakázek

Možnosti filtrování navrženého doporučovacího systému jsou omezené pouze na vlastnosti předmětu zakázky, lokality a předmětu podnikání zadavatele. Pro reálné užití systému by bylo vhodné po vzoru jiných současných systémů

doplnit možnosti filtrování podle dalších vlastností zakázek jako je jejich kategorické či časové zařazení, stav, cenový rozsah a podobně.

8.6 Klasifikátor CPV

Na základě analýzy datasetu v kapitole 4.1.3 uzavíram CPV klasifikaci dokumentů s odůvodněním nedostatečného datasetu.

V případě, že by se dostatečný dataset podařilo vytvořit, bylo by možné učit klasifikátor pro automatické rozpoznávání kategorie dokumentů a tudíž i zakázek.

8.7 Katalog produktů

V kapitole 3.3 pojednávám o možnostech využití katalogu produktů v systému.

Takový katalog se mi nepodařilo v rámci práce opatřit, ovšem v případě, že by se to podařilo, dalo by se s jeho pomocí získat další hodnotné informace o předmětu zakázek.

8.8 Vlastní model pro embedding dokumentů

V rámci zjednodušení kapitoly embeddingu dokumentů (4.2) jsem v práci použil předučené jazykové modely. Funkčnost modelů je ovšem závislá na po-vaze korpusu, na kterém jsou učené a proto by bylo vhodné v doméně veřejných zakázek naučit nebo alespoň vyladit model na korpusu, který by byl vytvořený z dokumentací samotných veřejných zakázek.

8.9 Ladění extrakce předmětu

Extrakce předmětu z textu je v oboru zpracování přirozeného jazyka problém sám o sobě, kterým se dlouhodobě zabývá mnoho odborníků.

Algoritmy extrakce předmětu navržené a implementované v této práci plní do jisté míry svůj účel, ovšem stále za přítomnosti nezanedbatelného množství falešných pozitivních i negativních nálezů. V tom smyslu by tedy mohla extrakce předmětu zakázek jít dále řešit jako celý samostatný projekt.

Závěr

V úvodu práce provádím rešerši současných systémů podporujících elektronické řízení a vyhledávání českých veřejných zakázek. Pojednávám o jejich vlastnostech a mezerách, na které se tato práce zaměřuje.

V rámci práce jsem navrhl a implementoval content-based doporučovací systém pro české veřejné zakázky. Doporučování je založeno na základě vybraných vlastností zakázek a uživatelském preferenčním profilu obsahujícím podobné vlastnosti.

Doporučování se uživateli přizpůsobuje podle jeho chování v aplikaci ve smyslu aktualizace jeho profilu informacemi zakázek, které si prohlédl.

Pro získávání vlastností jsem navrhl a implementoval proces extrakce informací z dokumentace veřejných zakázek a systémů poskytujících nejen data VZ.

Jako hlavní vlastnost zakázek jsem použil jejich předmět plnění, pro který jsem navrhl a implementoval algoritmy extrakce z textu dokumentace. Pro podchycení sémantické informace předmětu používám embedding textu. Volbu modelu pro embedding opíram o experimentální vyhodnocení současných state-of-the-art metod.

Projekt je implementovaný s modulární architekturou, kde jednotlivé komponenty jsem zdokumentoval pro možnost sestavení funkčního prototypu aplikace doporučovacího systému. Pro demonstraci jsem vytvořil jednoduchou webovou aplikaci, která umožňuje uživateli využít důležité vlastnosti systému.

V závěru práce pojednávám o možnostech rozšíření systému nutných pro jeho nasazení v reálném prostředí.

Celý projekt je vybudovaný za pomocí open-source technologií a pod open-source licencí je též vydaný.

Literatura

- [1] Weiss, N.: The Hitchhiker's Guide to Hierarchical Classification. *Towards Data Science [online]*, srpen 2019, [cit. 22. 4. 2020]. Dostupné z: <https://towardsdatascience.com/https-medium-com-noa-weiss-the-hitchhikers-guide-to-hierarchical-classification-f8428ea1e076>
- [2] Nooney, K.: Deep dive into multi-label classification..! (With detailed Case Study). *Towards Data Science [online]*, červen 2018, [cit. 22. 4. 2020]. Dostupné z: <https://towardsdatascience.com/journey-to-the-center-of-multi-label-classification-384c40229bff>
- [3] Palachy, S.: Document Embedding Techniques. *Towards Data Science [online]*, srpen 2019, [cit. 24. 2. 2020]. Dostupné z: <https://towardsdatascience.com/document-embedding-techniques-fed3e7a6a25d>
- [4] Mikolov, T.; Corrado, G.; Chen, K.; aj.: Efficient Estimation of Word Representations in Vector Space. 01 2013, s. 1–12.
- [5] Yang, Y.: Advances in Semantic Textual Similarity. *Google AI Blog [online]*, květen 2018, [cit. 24. 2. 2020]. Dostupné z: <https://ai.googleblog.com/2018/05/advances-in-semantic-textual-similarity.html>
- [6] Liang, X.: What is XLNet and why it outperforms BERT. *Towards Data Science [online]*, červen 2019, [cit. 24. 2. 2020]. Dostupné z: <https://towardsdatascience.com/what-is-xlnet-and-why-it-outperforms-bert-8d8fce710335>
- [7] Alammar, J.: The Illustrated BERT, ELMo, and co. (How NLP Cracked Transfer Learning). *jalammar.github.io [online]*, prosinec 2018, [cit. 24. 2. 2020]. Dostupné z: <http://jalammar.github.io/illustrated-bert/>

LITERATURA

- [8] Universal Dependencies [online]. [Cited 2020-04-11]. Dostupné z: <https://universaldependencies.org/>
- [9] Ministerstvo pro místní rozvoj, Odbor elektronizace veřejných zakázek: Výroční zpráva o stavu veřejných zakázek v České republice za rok 2018. *Výroční zprávy o stavu veřejných zakázek [online]*, květen 2019. Dostupné z: http://www.portal-vz.cz/getmedia/1a3cd915-5aea-4f1f-8a41-90565a2efe36/Vyrocní-zpráva-o-stavu-verejných-zakazek-v-Ceske-Republice-za-rok-2018_f.pdf
- [10] ČESKÉ ZAKÁZKY.CZ: Souhrnná statistika portálu České Zakázky.cz. *Statistika [online]*, [cit. 11. 5. 2020], url =.
- [11] Transparency International, Česká republika: Férové zadávání veřejných zakázek: Příručka pro zadavatele. *Publikace a analýzy [online]*, 2019. Dostupné z: <https://www.transparency.cz/wp-content/uploads/2019/03/F%C3%A9rov%C3%A9-%C5%99ejn%C3%A9-zak%C3%A1zky-v-ESI-fondech-P%C5%99%C3%ADru%C4%8Dka-pro-zadavatele.pdf>
- [12] Veřejná zakázka. *Wikipedie [online]*, [cit. 9. 5. 2020]. Dostupné z: https://cs.wikipedia.org/wiki/Ve%C5%99ejn%C3%A1_zak%C3%A1zka
- [13] Portál pro vhodné uveřejnění [online]. [Cited 2020-5-9]. Dostupné z: <https://www.vhodne-uverejneni.cz>
- [14] Tenderman [online]. [Cited 2020-5-9]. Dostupné z: <https://tenderman.cz/>
- [15] Hlídač státu [online]. [Cited 2020-04-22]. Dostupné z: <https://www.alza.cz/hlidac-statu/v15308.htm>
- [16] Otevřený software. *Wikipedie [online]*, [cit. 10. 5. 2020], url =.
- [17] Portál pro vhodné uveřejnění [online]. [Cited 2020-04-20]. Dostupné z: https://www.vhodne-uverejneni.cz/caste-dotazy#faq_6
- [18] Profil zadavatelů dle zákona č.134/2016 Sb. [online]. [Cited 2020-04-20]. Dostupné z: http://www.isvz.cz/ISVZ/VZ/ProfilyZadavatelu_134_2016.aspx
- [19] Evropská komise: Nařízení komise (ES) č. 213/2008. 2007.
- [20] Číselníky a klasifikace veřejných zakázek [online]. [Cited 2020-04-21]. Dostupné z: http://www.isvz.cz/ISVZ/Ciselniky/ISVZ_klasifikace_ciselniky.aspx
- [21] Heureka [online]. [Cited 2020-04-21]. Dostupné z: <https://www.heureka.cz/>

- [22] Administrativní registr ekonomických subjektů [online]. [Cited 2020-04-21]. Dostupné z: <https://wwwinfo.mfcr.cz/ares/>
- [23] Geokódování [online]. [Cited 2020-04-21]. Dostupné z: <https://api.mapy.cz/view?page=geocoding>
- [24] Mikolov, T.; Sutskever, I.; Chen, K.; aj.: Distributed Representations of Words and Phrases and their Compositionality. *Advances in Neural Information Processing Systems*, ročník 26, 10 2013.
- [25] Kenter, T.; Borisov, A.; de Rijke, M.: Siamese CBOW: Optimizing Word Embeddings for Sentence Representations. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Berlin, Germany: Association for Computational Linguistics, Srpen 2016, s. 941–951, doi:10.18653/v1/P16-1089. Dostupné z: <https://www.aclweb.org/anthology/P16-1089>
- [26] Le, Q.; Mikolov, T.: Distributed Representations of Sentences and Documents. *31st International Conference on Machine Learning, ICML 2014*, ročník 4, 05 2014.
- [27] Pennington, J.; Socher, R.; Manning, C.: Glove: Global Vectors for Word Representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Doha, Qatar: Association for Computational Linguistics, Říjen 2014, s. 1532–1543, doi:10.3115/v1/D14-1162. Dostupné z: <https://www.aclweb.org/anthology/D14-1162>
- [28] Ganegedara, T.: Intuitive Guide to Understanding GloVe Embeddings. *Towards Data Science [online]*, květen 2019, [cit. 24. 2. 2020]. Dostupné z: <https://towardsdatascience.com/light-on-math-ml-intuitive-guide-to-understanding-glove-embeddings-b13b4f19c010>
- [29] Hill, F.; Cho, K.; Korhonen, A.: Learning Distributed Representations of Sentences from Unlabelled Data. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, San Diego, California: Association for Computational Linguistics, Červen 2016, s. 1367–1377, doi:10.18653/v1/N16-1162. Dostupné z: <https://www.aclweb.org/anthology/N16-1162>
- [30] Pagliardini, M.; Gupta, P.; Jaggi, M.: Unsupervised Learning of Sentence Embeddings Using Compositional n-Gram Features. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume*

LITERATURA

- 1 (*Long Papers*), New Orleans, Louisiana: Association for Computational Linguistics, Červen 2018, s. 528–540, doi:10.18653/v1/N18-1049. Dostupné z: <https://www.aclweb.org/anthology/N18-1049>
- [31] Logeswaran, L.; Lee, H.: An efficient framework for learning sentence representations. 03 2018.
 - [32] Kusner, M.; Sun, Y.; Kolkin, N.; aj.: From word embeddings to document distances. *Proceedings of the 32nd International Conference on Machine Learning (ICML 2015)*, 01 2015: s. 957–966.
 - [33] Wu, L.; Yen, I.; Xu, F.; aj.: D2KE: From Distance to Kernel and Embedding. 02 2018.
 - [34] Peters, M.; Neumann, M.; Iyyer, M.; aj.: Deep Contextualized Word Representations. 01 2018, s. 2227–2237, doi:10.18653/v1/N18-1202.
 - [35] Howard, J.; Ruder, S.: Universal Language Model Fine-tuning for Text Classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Melbourne, Australia: Association for Computational Linguistics, Červenec 2018, s. 328–339, doi:10.18653/v1/P18-1031. Dostupné z: <https://www.aclweb.org/anthology/P18-1031>
 - [36] Ghelani, S.: From Word Embeddings to Pretrained Language Models — A New Age in NLP — Part 2. *Towards Data Science [online]*, květen 2019, [cit. 24. 2. 2020]. Dostupné z: <https://towardsdatascience.com/from-word-embeddings-to-pretrained-language-models-a-new-age-in-nlp-part-2-e9af9a0bdcd9>
 - [37] Vaswani, A.; Shazeer, N.; Parmar, N.; aj.: Attention Is All You Need. 06 2017.
 - [38] Cer, D.; Yang, Y.; Kong, S.-y.; aj.: Universal Sentence Encoder. 03 2018.
 - [39] Radford Alec, T. S. I. S., Karthik Narasimhan: Improving Language Understanding by Generative Pre-Training. 2018.
 - [40] Radford Alec, R. C. D. L. D. A. I. S., Jeffrey Wu: Language Models are Unsupervised Multitask Learners. 2019.
 - [41] Devlin, J.; Chang, M.-W.; Lee, K.; aj.: BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. říjen 2018, doi:10.18653/v1/N19-1423. Dostupné z: <https://www.aclweb.org/anthology/N19-1423>

- [42] Horev, R.: BERT Explained: State of the art language model for NLP. *Towards Data Science [online]*, listopad 2018, [cit. 24. 2. 2020]. Dostupné z: <https://towardsdatascience.com/bert-explained-state-of-the-art-language-model-for-nlp-f8b21a9b6270>
- [43] Alammar, J.: The Illustrated Transformer. *jalammar.github.io [online]*, červen 2018, [cit. 24. 2. 2020]. Dostupné z: <https://jalammar.github.io/illustrated-transformer/>
- [44] Reimers, N.; Gurevych, I.: Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. 01 2019, s. 3973–3983, doi:10.18653/v1/D19-1410.
- [45] Yang, Z.; Dai, Z.; Yang, Y.; aj.: XLNet: Generalized Autoregressive Pre-training for Language Understanding. 2019, 1906.08237.
- [46] Dai, Z.; Yang, Z.; Yang, Y.; aj.: Transformer-XL: Attentive Language Models beyond a Fixed-Length Context. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, Florence, Italy: Association for Computational Linguistics, Červenec 2019, s. 2978–2988, doi:10.18653/v1/P19-1285. Dostupné z: <https://www.aclweb.org/anthology/P19-1285>
- [47] Suryavansh, M.: 2019 — Year of BERT and Transformer. *Towards Data Science [online]*, leden 2020, [cit. 24. 2. 2020]. Dostupné z: <https://towardsdatascience.com/2019-year-of-bert-and-transformer-f200b53d05b9>
- [48] Lan, Z.; Chen, M.; Goodman, S.; aj.: ALBERT: A Lite BERT for Self-supervised Learning of Language Representations. 2019, 1909.11942.
- [49] Yang, Y.; Cer, D.; Ahmad, A.; aj.: Multilingual Universal Sentence Encoder for Semantic Retrieval. 2019, 1907.04307.
- [50] Eisenschlos, J.; Ruder, S.; Czaplak, P.; aj.: MultiFiT: Efficient Multilingual Language Model Fine-tuning. 2019, 1909.04761.
- [51] Hořenovská, K.: An evaluation of Czech word embeddings. In *Proceedings of the 22nd Nordic Conference on Computational Linguistics*, Turku, Finland: Linköping University Electronic Press, Září–Říjen 2019, s. 65–75. Dostupné z: <https://www.aclweb.org/anthology/W19-6107>
- [52] Svoboda, L.; Brychcín, T.: New word analogy corpus for exploring embeddings of Czech words. 2016, 1608.00789.
- [53] Konopík, M.; Pražák, O.; Steinberger, D.: Czech Dataset for Semantic Similarity and Relatedness. In *Proceedings of the International Conference Recent Advances in Natural Language Processing, RANLP 2017*, Varna,

LITERATURA

- Bulgaria: INCOMA Ltd., Září 2017, s. 401–406, doi:10.26615/978-954-452-049-6_053. Dostupné z: https://doi.org/10.26615/978-954-452-049-6_053
- [54] Svoboda, L.; Brychcín, T.: Czech Dataset for Semantic Textual Similarity. In *Text, Speech, and Dialogue*, editace P. Sojka; A. Horák; I. Kopeček; K. Pala, Cham: Springer International Publishing, 2018, ISBN 978-3-030-00794-2, s. 213–221.
 - [55] Conneau, A.; Kiela, D.: SentEval: An Evaluation Toolkit for Universal Sentence Representations. 2018, 1803.05449.
 - [56] Prague Dependency Treebank [online]. [Cited 2020-04-12]. Dostupné z: <http://ufal.mff.cuni.cz/pdt>
 - [57] Treex [online]. [Cited 2020-04-12]. Dostupné z: <http://ufal.mff.cuni.cz/treex>
 - [58] UDPipe [online]. [Cited 2020-04-12]. Dostupné z: <http://ufal.mff.cuni.cz/udpipe>
 - [59] Udapi [online]. [Cited 2020-04-13]. Dostupné z: <https://udapi.github.io/>
 - [60] Zeman, D.; Nivre, J.; Abrams, M.; aj.: Universal Dependencies 2.5. 2019, LINDAT/CLARIAH-CZ digital library at the Institute of Formal and Applied Linguistics (ÚFAL), Faculty of Mathematics and Physics, Charles University. Dostupné z: <http://hdl.handle.net/11234/1-3105>
 - [61] Mayank, M.: String similarity — the basic know your algorithms guide! *ITNEXT* [online], únor 2019, [cit. 27. 4. 2020]. Dostupné z: <https://itnext.io/string-similarity-the-basic-know-your-algorithms-guide-3de3d7346227>
 - [62] Emmery, C.: Euclidean vs. Cosine Distance. *cmry.github.io* [online], březen 2017, [cit. 27. 4. 2020]. Dostupné z: <https://cmry.github.io/notes/euclidean-v-cosine>
 - [63] Dey, S.: Distance Matrix Vectorization Trick. *Medium* [online], srpen 2016, [cit. 7. 5. 2020]. Dostupné z: <https://medium.com/@souravdey/12-distance-matrix-vectorization-trick-26aa3247ac6c>
 - [64] Rocca, B.: Introduction to recommender systems. *Towards Data Science* [online], červen 2019, [cit. 28. 2. 2020]. Dostupné z: <https://towardsdatascience.com/introduction-to-recommender-systems-6c66cf15ada>

Seznam použitých zkrátek

API Application Programming Interface

ARES Administrativní registr ekonomických subjektů

BERT Bidirectional Encoder Representations from Transformers

BOW Bag-of-words

CBOW Continuous Bag-of-words

CPV Common Procurement Vocabulary

ČR Česká republika

ECZ Evidenční číslo zakázky

ELMo Embeddings from Language Model

GB Gigabyte

GPS Global Positioning System

GPT Generative Pre-Training Transformer

HDP Hrubý domácí produkt

IČO Identifikační číslo osoby

ISVZ Informační systém o veřejných zakázkách

LDA Latent Dirichlet Allocation

LSTM Long Short-term Memory

NLP Natural Language Processing

PDT Prague Dependency Treebank

A. SEZNAM POUŽITÝCH ZKRATEK

PLSI Probabilistic Latent Semantic Indexing

POS Part-of-speech

STS Semantic Textual Similarity

tf-idf term frequency – inverse document frequency

UD Universal Dependencies

ULMFiT Universal Language Model Fine-tuning

USE Universal Sentence Encoder

VZ Veřejná zakázka

WMD Word Mover's Distance

WME Word Mover's Embedding

Obsah přiloženého CD

```
readme.txt ..... stručný popis obsahu CD
├── exe ..... adresář se spustitelnou formou implementace
├── src
│   ├── impl ..... zdrojové kódy implementace
│   └── thesis ..... zdrojová forma práce ve formátu LATEX
└── text
    ├── thesis.pdf ..... text práce ve formátu PDF
    └── thesis.ps ..... text práce ve formátu PS
```