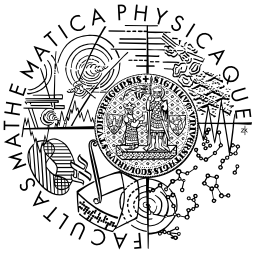


MATEMATICKO-FYZIKÁLNÍ FAKULTA
Univerzita Karlova

DIPLOMOVÁ PRÁCE

Bc. Jakub Maroušek

Automatická klasifikace smluv pro portál HlidacSmluv.cz

Katedra softwarového inženýrství

Vedoucí diplomové práce: doc. Mgr. Martin Nečaský, Ph.D.

Studijní program: Informatika

Studijní obor: Softwarové systémy

Praha 2020

Prohlašuji, že jsem tuto diplomovou práci vypracoval(a) samostatně a výhradně s použitím citovaných pramenů, literatury a dalších odborných zdrojů. Tato práce nebyla využita k získání jiného nebo stejného titulu.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona v platném znění, zejména skutečnost, že Univerzita Karlova má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle §60 odst. 1 autorského zákona.

V ..... dne ....

Podpis autora

Rád bych poděkoval Michalu Bláhovi za možnost spolupracovat na vývoji projektu se společenským přínosem. Děkuji za poskytnutí zdrojových dat a mnoho rad a doporučení při jejich zpracování.

Poděkování rovněž patří doc. Mgr. Martinu Nečaskému, Ph.D., vedoucímu práce, za konzultace týkající se klasifikace dokumentů a textu diplomové práce. Mgr. Barboře Vidové Hladké, Ph.D. děkuji za pomoc při výběru metod strojového učení.

V neposlední řadě děkuji Bc. Martinu Marešovi za cenné připomínky týkající se strojového učení. Velké poděkování patří mé rodině za podporu při studiu.

Název práce: Automatická klasifikace smluv pro portál HlidacSmluv.cz

Autor: Bc. Jakub Maroušek

Katedra: Katedra softwarového inženýrství

Vedoucí diplomové práce: doc. Mgr. Martin Nečaský, Ph.D., katedra softwarového inženýrství

Abstrakt: Registr smluv je veřejná databáze obsahující smlouvy uzavřené institucemi veřejné správy. Vzhledem k množství publikovaných dokumentů je však analýza dat problematická. Cílem práce je za použití metod strojového učení naleznout postup pro rozdělení smluv do kategorií podle oblastí (realitní služby, stavitelství a podobně) a tento postup implementovat pro použití na webovém portálu Hlídač státu. Komplikaci představuje velké množství kategorií a fakt, že není k dispozici žádná sada již označených smluv.

Klíčová slova: e-government, strojové učení, transfer learning, multilabel klasifikace, klasifikace dokumentů

Title: Automated contract classification for portal HlidacSmluv.cz

Author: Bc. Jakub Maroušek

Department: Department of Software Engineering

Supervisor: doc. Mgr. Martin Nečaský, Ph.D., department of Software Engineering

Abstract: The Contracts Register is a public database containing contracts concluded by public institutions. Due to the number of documents in the database, data analysis is problematic. The objective of this thesis is to find a machine learning approach for sorting the contracts into categories by their area of interest (real estate services, construction, etc.) and implement the approach for usage on the web portal Hlídač státu. A large number of categories and a lack of a tagged dataset of contracts complicate the solution.

Keywords: e-government, machine learning, transfer learning, multi-label classification, document classification

Obsah

Úvod

E-government je termín označující zavádění digitálních technologií do funkcí a procesů ve státní správě. Tento trend přináší nejen zjednodušení komunikace se státními institucemi, ale také umožňuje nad nimi provádět transparentnější dohled. Typickým způsobem, jak orgány veřejné správy zajišťují transparentnost, je zveřejňováním informací o chodu instituce. Využití digitálních technologií umožňuje tyto informace snadno získávat a analyzovat.

V České republice například mají organizace veřejné správy povinnost vkládat veškeré uzavřené smlouvy nad určitou částku do *registru smluv*. Tento digitální registr je veřejný. Pro každou smlouvu jsou zde dostupné nejen informace o předmětu smlouvy, smluvních stranách a částce, ale také veškeré soubory s texty smlouvy. Registr představuje významný nástroj pro zlepšení transparentnosti; jeho cílem je omezit možnost plýtvání a korupce kvůli uzavírání nevýhodných smluv.

Ačkoliv používání registru a zveřejňování dat do něj je právně vynutitelné, tento fakt ještě nezaručuje možnost snadné analýzy dat a přehledného vyhledávání. Pro tyto účely byl v soukromé sféře vyvinut webový portál Hlídač smluv, jehož cílem bylo umožnit snadné a přehledné zpracování dat z registru. Později, po zkombinování dalších veřejně přístupných registrů a databází, vznikl větší projekt Hlídač státu[1]. Projekt například umožňuje plnohodnotné fulltextové vyhledávání v textech smluv; toto v původním registru smluv není možné, protože mnoho přiložených souborů jsou fotografie s textem.

Registr smluv obsahuje dokumenty z mnoha různých oblastí, ať už jde o realitní služby, IT projekty, nebo stavitelství. V současných datech však neexistuje žádný snadný způsob, jak smlouvy do těchto oblastí zařadit. Smluv v registru je přes 400 tisíc (a jejich počet stále roste), takže roztřídění smluv by velmi zvýšilo přehlednost a usnadnilo hledání. Další motivací je však konkrétní projekt Hlídače státu *K-index*. Cílem tohoto projektu je přiřadit každé státní instituci index vyjadřující míru podezření na nezákonné jednání. Silným ukazatelem potenciálně korupčního prostředí je situace, kdy zakázky v určité oblasti – například IT – jsou zadávány pouze jediné soukromé firmě. K implementaci tohoto pravidla je však nutné mít smlouvy rozdělené do kategorií.

Cílem této práce je analyzovat dostupná data o smlouvách, využít metody strojového učení a navrhnout systém, který umožní na základě dostupných dat klasifikovat smlouvy. Implementaci tohoto systému je třeba nasadit na serverech Hlídače státu.

Seznam kategorií smluv byl dodán jako vstup; celkem je jich 104. Specifickou vlastností problému je toto velké množství kategorií. Kategorie navíc nejsou disjunktní: jedna smlouva může být zařazená do více kategorií současně. Další komplikací je fakt, že nemáme k dispozici sadu smluv se správně přiřazenými kategoriemi, kterou bychom mohli využít k trénování klasifikátoru.

Hlavním přínosem práce je porovnání výkonnosti klasifikace textů při situaci, kdy trénovací a testovací data nepochází ze stejné domény, a kdy kategorií k přiřazení je velké množství. Díky tomuto porovnání jsme nalezli nejvýhodnější klasifikátor, který byl následně použit v praxi.

[1] `https://www.hlidacstatu.cz/`

Struktura práce je následující. Kapitola 1 se věnuje e-governmentu v Česku a jeho mezinárodnímu srovnání. Obsahuje informace o registrech smluv a veřejných zakázek, popisuje jejich obsah a strukturu záznamů. Rovněž blíže představuje projekt Hlídač státu a konkrétní metody pro vylepšení analýzy dokumentů. Kapitola 2 specifikuje konkrétní zadání a upřesňuje požadavky na výsledný systém. Kapitola 3 je věnována analýze dostupných datových sad. V kapitole 4 diskutujeme postupy, kterými je možné problém řešit; zde také vybíráme metody klasifikace k otestování. Experimenty, metriky jejich posuzování a popis výsledků nalezneme v kapitole 5. Poslední krok, implementaci a nasazení na server Hlídače státu, je popsán v kapitole 6.

1. E-government a související projekty

V této úvodní kapitole se budeme věnovat e-governmentu, využití digitálních technologií pro zlepšení funkce veřejné správy. Srovnáme situaci vývoje e-governmentu v České republice a jiných státech. Také si představíme projekty, které si dávají za cíl zlepšit dohled nad funkcí státní správy pomocí informačních technologií – jedním z nich je i portál Hlídač státu, do něhož tato práce přináší novou funkcionalitu.

1.1 Elektronizace a e-government

Obecný trend zavádění informačních technologií do všemožných sfér lidského života se promítá i do provozování státní správ v jednotlivých zemích. Podle Hai a Jeong [25] se za *e-government* souhrnně označují „způsoby využití informačních technologií a digitálních komunikačních technologií pro zlepšení efektivity ve veřejném sektoru“. Podle Štědroně [59] je e-government definován jako „série procesů, umožňující výkon veřejné správy a uplatňování občanských práv a povinností fyzických a právnických osob, realizovaných elektronickými prostředky“. Z uvedených definic vyplývá několik druhů aktivit, ve kterých se tato vyšší efektivita projeví:

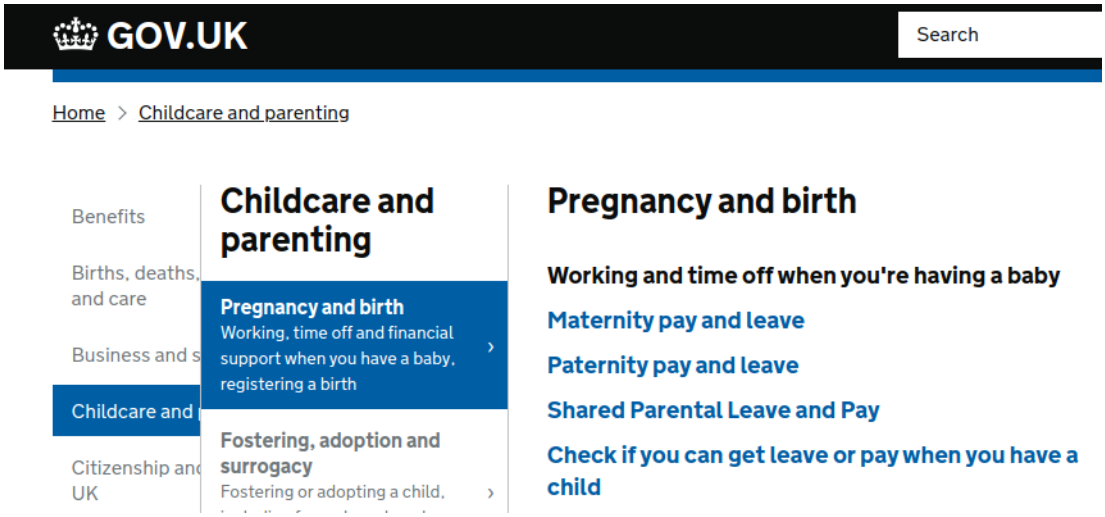
- získávání informací od institucí veřejné správy (například postup pro získání stavebního povolení),
- komunikace s institucemi (může být formální, jako například podání daňového přiznání, nebo méně formální, třeba podnět pro vylepšení činnosti),
- dohled nad státní správou, prevence korupce.

E-government souvisí se snahou přestat vnímat vztah mezi občanem a státem ve smyslu *podřízeného a vrchnosti* a posunout ho do roviny *zákazníka a poskytovatele*. Tuto vizi popisuje například Michal Bláha [63]. Stát (a úředníci státní správy) by neměl být vnímán jako nadřazená jednotka, nýbrž jako ten, kdo občanům poskytuje nějaké služby.

1.2 Srovnání kvality e-governmentu

Možnosti využití e-governmentu lze nejlépe přiblížit na příkladech zemí, kterým se jej daří úspěšně zavádět do praxe. Dobrým zdrojem k porovnání míry úspěšnosti je například United Nations E-Government Survey, každoroční výzkum Odboru pro ekonomické a sociální otázky Organizace spojených národů. Nejnovější vydání pro rok 2018 [51] u jednotlivých států posuzuje například míru přístupu občanů k informačním technologiím, jejich znalosti výpočetní techniky, nabídku služeb v jednotlivých oblastech státní správy a bezpečnostní politiky.

Jako příklad státu, který úspěšně zavádí služby e-governmentu, můžeme uvést Velkou Británii (v roce 2018 na 4. pozici ze všech států). Centrální portál na



Obrázek 1.1: Příklad hierarchického hledání informací na portálu *gov.uk*

adrese `https://www.gov.uk/` zahrnuje informace ze všech ministerstev země. Obrázek 1.1 ukazuje hierarchické rozdělení služeb na webu. Primární funkcí *gov.uk* není dynamická komunikace, ale vyhledávání informací. Veškerý text na webu musí být psaný podle zásad *Simple English*, tedy tak jednoduše, jak to je možné; je třeba se vyhnout složitým slovním spojením nebo komplikované terminologii.

I přes určité pokroky v zavádění e-governmentu do praxe se Česká Republika v UN E-Government Survey v žebříčku států propadá. V roce 2018 se umístila na 54. místě, což je od roku 2017 propad o čtyři příčky. [51]

Jak plyne ze zpráv Národního kontrolního úřadu [1], vysoutěžené zakázky na informační systémy jsou často předražené. Podobně drahý je také provoz systémů. Jako příklad poslouží projekt Datových schránek, kdy provozovateli na začátku fungování systému bylo placeno 15,04 Kč za jednu zprávu. [48] Systémy jsou často nepřívětivé. V případě Datových schránek bylo dlouhou dobu potřeba instalovat rozšíření do webového prohlížeče, což velmi omezovalo rozsah softwaru na straně koncového uživatele, kde systém mohl fungovat.

1.3 Projekty na podporu dohledové funkce e-governmentu

Z hlediska kontroly funkce státní správy v Česku je zásadní zákon č. 106/1999 Sb, o svobodném přístupu k informacím. Specifikuje informace, které o sobě musí zveřejňovat každá státní instituce nebo orgán. Zákon se dále zabývá povinností zveřejnit informace, pokud státní orgán dostane žádost o zveřejnění, a vymezuje kategorie informací, které nemusí být zveřejněné.

Téma této práce se dotýká e-governmentu, konkrétně však jeho dohledové funkce nad státní správou. Popíšeme si proto tři projekty, které si kladou za cíl zveřejňovat data o funkci státní správy za účelem prevence korupce a předražených zakázek. První dva projekty mají svůj původ přímo ve veřejné správě – jde o registr smluv a registr veřejných zakázek. Třetí projektem je portál Hlídač státu, který z registrů čerpá data a zpracovává je tak, aby je mohla studovat široká veřejnost. Tento projekt má svůj původ v soukromé sféře.

1.3.1 Registr smluv

Použití e-governmentu pro funkci dohledu nad státní správou zaznamenalo v Česku pokrok zvláště při zavedení Registru smluv v roce 2016. [2] Tento zákon ukládá všem veřejným institucím povinnost zveřejňovat soukromoprávní smlouvy a smlouvy o poskytnutí dotace v hodnotě nad 50 tisíc korun bez DPH. Hlavní význam registru je ve zvyšování transparentnosti chodu státní správy. Je možné navzájem srovnávat vynaložené prostředky jednotlivých institucí a identitu protistran a poukazovat na nevýhodné zakázky.

Inspirací pro založení registru smluv bylo Slovensko. Zákon o zveřejňování smluv tu byl přijat již v roce 2010; Slovensko se stalo první evropskou zemí, kde smlouvy musely být zveřejněné ze zákona. Podle výzkumu Transparency International [69] od doby zavedení registru 11 procent obyvatel do registru nahlédlo alespoň jednou. Dvě procenta populace (asi 90 tisíc lidí) pak registr zkoumá pravidelně. Výzkum udává konkrétní případy, kdy zveřejněná smlouva s kontroverzní vlastností (vysoká cena, pochybný dodavatel, předmět smlouvy) si získala zájem médií. Dotčená státní instituce se dostala pod mediálním tlak a někdy byla přinucena změnit podmínky smlouvy nebo ji zrušit.

Přístup k registru smluv je zajištěn přes webové rozhraní.[1]. Zde je možné buď smlouvy buď přímo vyhledávat, nebo si stáhnout otevřená data ve formátu XML, v nichž je uveden odkaz na textové přílohy.

Mezi výjimky, kdy se smlouva do registru ukládat nemusí, patří činnost zpravodajských a bezpečnostních služeb, místo plnění mimo území Česka, nebo souvislost s nějakou mimořádnou událostí ohrožující život nebo majetek. Ve smlouvě je možné „začernit“ (znečitelnit) osobní údaje fyzických osob a obchodní tajemství. Mezi obchodní tajemství může patřit i identifikace smluvních stran. Smlouva, která musí být ze zákona uvedená v registru, ale není do něj do 30 dnů od dojednání odeslána, je považována za neplatnou.

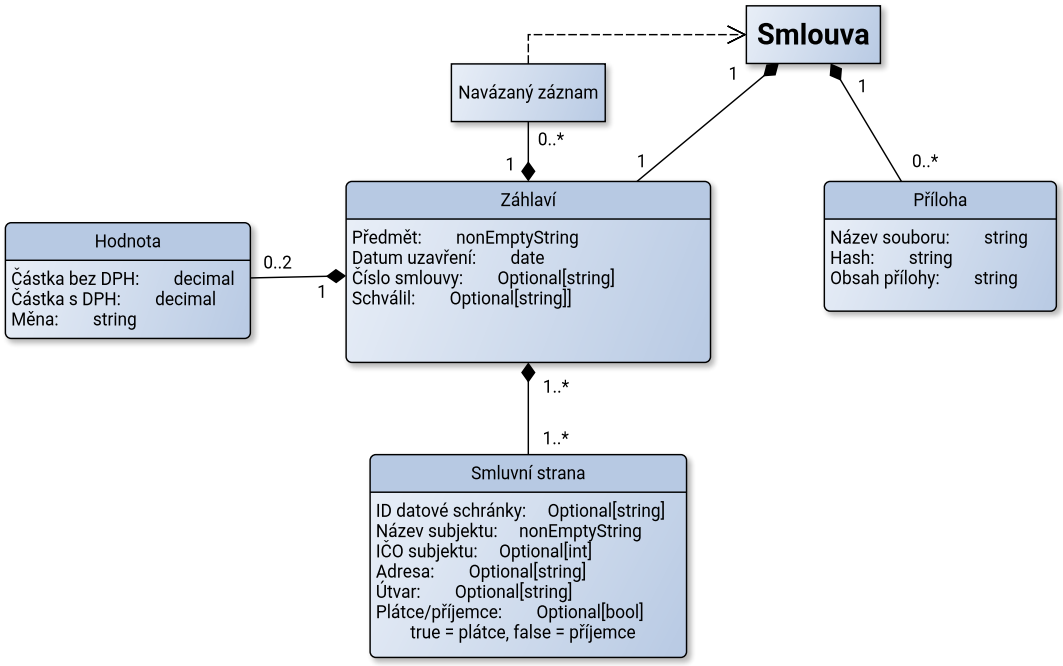
Struktura záznamu smlouvy

Podrobná struktura záznamu smlouvy je znázorněna na obrázku 1.2. Každá smlouva se skládá ze *záhlaví*, obsahujícího metadata o smlouvě, a několika *příloh*, souborů s textem smlouvy.

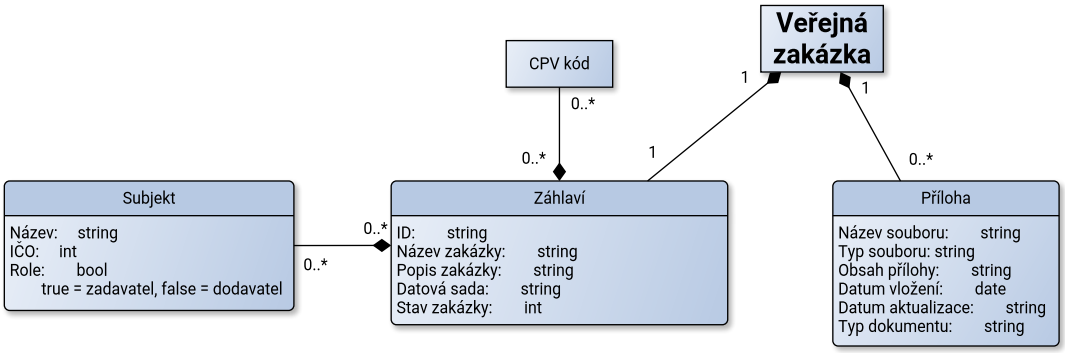
V záhlaví se nachází dva identifikátory: *identifikátor verze smlouvy* je unikátní pro každý záznam, *identifikátor smlouvy* je společný pro ty záznamy, které se fakticky týkají jediné smlouvy. Povinnou součástí záhlaví je dále pouze předmět a datum uzavření; další položky, jako například hodnota smlouvy nebo identifikace smluvních stran, mohou v oprávněných případech zůstat nevyplněné.

Smluvní strany mají uvedené informace v záznamu *smluvní strana*. Pro každou smlouvu musí být povinně uvedena pouze jedna smluvní strana, a to veřejná instituce, které se smlouva týká. Veškeré položky smluvních stran (datová schránka, adresa, IČO) jsou opět nepovinné, s výjimkou názvu každé smluvní strany. Hodnota smlouvy je uvedena s DPH a bez DPH. Ke smlouvě může být připojena hodnota v českých korunách a v zahraniční měně. Každá smlouva může být také pomocí položky *navázaný záznam* spojena s jinými smlouvami, s nimiž fakticky souvisí.

[1] `https://smlouvy.gov.cz/`



Obrázek 1.2: UML schéma datového záznamu smlouvy



Obrázek 1.3: UML schéma datového záznamu veřejné zakázky

V přílohách pak musí být vložen text smlouvy, neboli podle zákona *elektronický obraz textového obsahu smlouvy v otevřeném a strojově čitelném formátu.* Podle metodiky registru jsou podporovány formáty PDF, DOC, DOCX, RTF, ODT a TXT [44], v praxi je ale možné najít i přílohy v jiných formátech, jako třeba XML.

Problematické záznamy v registru

Mezi sporné body registru patří otázka obchodního tajemství. Během procesu schvalování zákona o registru smluv docházelo k politickým bojům ohledně toho, co všechno musí být pro smlouvy zveřejněné, a které státní instituce by měly být úplně osvobozené od povinnosti smlouvy zveřejňovat. V praxi se jednalo například o Českou televizi, Český rozhlas nebo Budějovický Budvar – tedy podniky, které se pohybují v konkurenčním prostředí. Velké množství smluv v registru, za první rok přes 20 procent, má skrytou cenu, přes 8 procent má skrytého dodavatele.

Nedá se však jednoduše odhadnout, které z cen nebo dodavatelů jsou zakryti oprávněně. Například Budějovický Budvar má zakryté ceny na 99,5 procentech smluv. [11]

Dalším problémem je fakt, že provozovatel systému, Ministerstvo vnitra, neodpovídá za správnost dat a neprovádí kontrolu metadat. [9] Velké množství textů smluv například tvoří naskenované fotografie (vložené například v PDF), které nejsou strojově čitelné.

1.3.2 Registr veřejných zakázek

Jako veřejná zakázka se označuje postup, kterým si státní instituce zajišťují své potřeby – výroba zboží, provedení prací a služeb a podobně – prostřednictvím soukromých institucí, mezi nimiž může probíhat konkurenční soutěž. Cílem zadávání veřejných zakázek je ekonomické využívání veřejných prostředků a menší pravděpodobnost korupce. K tomu však je třeba stanovit co nejtransparentnější pravidla pro postupy při vyřizování zakázek. [27]

Na rozdíl od centrálního registru smluv neexistuje v České republice jednotný registr veřejných zakázek. Zakázky s cenou nad 1 milion korun (stavební zakázky nad 3 miliony korun) musí být uveřejněny ve *Věstníku veřejných zakázek*, který je centralizovaný a veřejné zakázky nad určitou částku podle směrnice EU jsou z něj exportovány do evropského věstníku TED. Ostatní zakázky jsou uveřejněny na *Profilech zadavatelů*, speciálních webových stránkách umožňujících zveřejnění veřejných zakázek. Každý potenciální zadavatel ale používá vlastní profil, typicky zveřejněný na svém webovém portálu. Vyhledávání je tedy velmi problematické. V praxi existují nástroje, které shromažďují informace o veřejných zakázkách ze všech zdrojů. Jsou obvykle placené; portál Hlídač státu ale tuto službu poskytuje také.

Struktura záznamu veřejné zakázky

Na rozdíl od smluv nemají veřejné zakázky jednotný datový formát a ani není přesně specifikované, která datová pole jsou povinná. Platí, že zakázky uveřejněné ve Věstníku veřejných zakázek mají striktnější pravidla, například musí mít uvedené CPV kódy. Formát, který zde popíšeme, vychází ze struktury, která je používána v databázi portálu Hlídač státu. Tato struktura je znázorněna na obrázku 1.3.

Podobně jako data smlouvy, i data veřejné zakázky jsou rozdělena do záhlaví a příloh. Unikátním identifikátorem je pole *ID*, základní informace o zakázce je možné se dozvědět z *názvu zakázky* a *popisu zakázky*; popis zakázky je ale nejčastěji prázdný. V poli *datová sada* se nachází identifikátor zdroje, z něhož byla veřejná zakázka získána (Věstník veřejných zakázek nebo profil konkrétního zadavatele). Číslo v poli *stav zakázky* značí, zda zadavatel čeká na nabídky od dodavatelů, případně zda zakázka již byla vysoutěžena, nebo byla zrušena. Na rozdíl od smluv se informace o potenciálních smluvních stranách skládají pouze z názvu subjektu a identifikačního čísla. Součástí záhlaví je také seznam CPV kodů, může se však stát, že je tento seznam prázdný.

Struktura záznamu přílohy je podobná odpovídající struktuře pro smlouvu. *Název souboru* obsahuje příponu označující typ dokumentu, ačkoliv pro typ je

určené samostatné pole. Samotná příloha je dostupná v poli *obsah přílohy*. Obsahem pole *typ dokumentu* je sémantický význam přílohy, například „Specifikace zakázky“, *Nabídka dodavatele* a podobně.

CPV kódy

Pro snazší orientaci v seznamech zakázek a pro možnost sledování zakázek podle oboru (kategorie) byly evropskou směrnicí zavedeny *CPV kódy (Common Procurement Vocabulary)*, hierarchický systém kategorií. Kód jakéhokoliv produktu se skládá až z devíti číslic: první dvě číslice určují *obor*, nejobecnější úroveň hierarchie; další číslice určují podřazené kategorie. Čím více číslic kód obsahuje, tím konkrétnější kategorii reprezentuje. Následuje příklad hierarchie systému pro jeden obor:

- 18: Oděvy, obuv, brašnářské výrobky a doplňky
  - 181: Zaměstnanecké oděvy, speciální pracovní oděvy a oděvní doplňky
  - 182: Svrchní oděvy
  - 183: Prádlo a oděvní součásti
    - 1831: Spodní prádlo
    - 1832: Podprsenky, korzety, podvazky a podobné výrobky
    - 1833: Trička s krátkým rukávem (T-shirts) a košile
    - ...
  - ...

Ukážeme si také příklady některých oborů. Nižší čísla oborů se týkají obchodu se základními, nezpracovanými surovinami (zemědělské výrobky, dřevo, uhlí, rudy kovů atd.); o něco vyšší čísla se týkají výrobků (kancelářské stroje, telekomunikační zařízení, dopravní zařízení); ještě vyšší čísla jsou spojena se službami (stavební služby, dopravní služby, pojištění, právní služby).

- 03: Produkty zemědělství, hospodářské produkty, produkty akvakultury, lesnictví a související produkty
- 14: Produkty těžebního průmyslu, kovové suroviny a související produkty
- 16: Zemědělské stroje
- 22: Tiskařské výrobky a související produkty
- 24: Chemické výrobky
- 31: Elektrické strojní zařízení, přístroje, zařízení a spotřební materiál, osvětlení
- 44: Stavební konstrukce a materiály; pomocné výrobky pro konstrukce (mimo elektrické přístroje)
- 70: Realitní služby

Každý veřejná zakázka musí mít přiřazený *hlavní CPV kód*, který určuje kategorii co nejpřesněji, a případně několik *vedlejších CPV kódů*.

Existuje více než 9 400 CPV kódů. Do tohoto čísla počítáme i obecnější kódy, které se skládají z méně než devíti číslic.

1.3.3 Služba Hlídač státu

Předešlé sekce se věnovaly registrům státních smluv a veřejných zakázek. Ačkoliv jsou data v registrech komukoliv přístupná, tento fakt ještě nezajišťuje možnost komfortní analýzy ze strany uživatelů. Zmiňovali jsme, že velké množství textů smluv není strojově čitelné; veřejné zakázky jsou roztroušené na velkém množství webových portálů. Ale i v případě, že máme datové sady k dispozici, zde vzniká potřeba je analyzovat.

Hlídač státu je projektem, který vznikl v srpnu 2016 z původních projektů Hlídač smluv a Hlídač EET. Hlavním autorem projektů je Michal Bláha, český podnikatel v oblasti výpočetní techniky a e-governmentu. [8] Webový portál projektu je přístupný na doméně `https://www.hlidacstatu.cz/`. Cíle Hlídače státu jsou následující:

- přehledně a srozumitelně zveřejnit informace o hospodaření státu a samosprávy za použití veřejně přístupných registrů,
- tyto informace dále propojovat a obohacovat o data ze souvisejících databází,
- analyzovat informace a umožnit veřejnosti jejich snadnou analýzu,
- identifikovat zneužití moci v úřadech (korupce, předražené zakázky). [10]

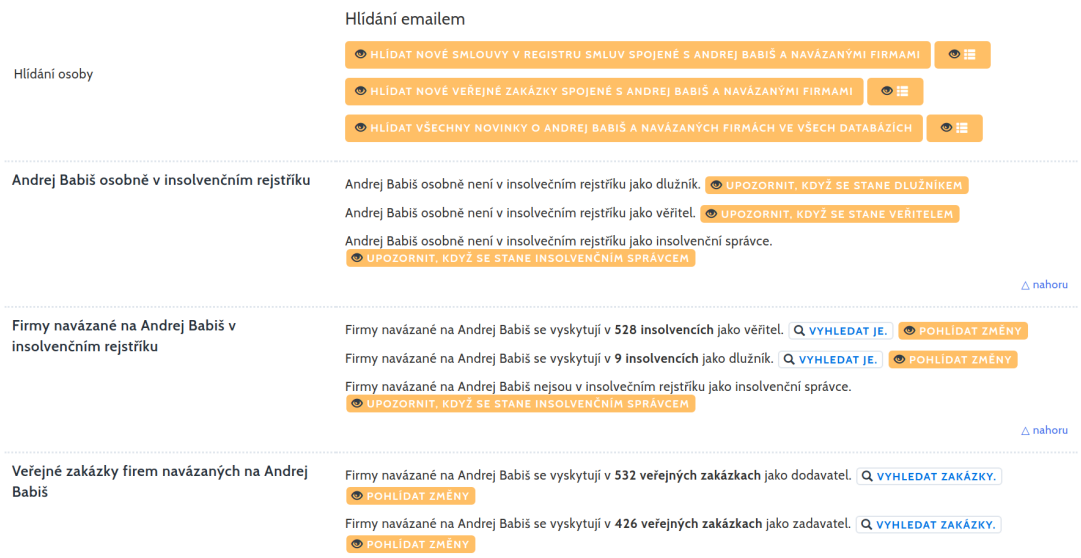
Oblasti zájmu webu

Logicky je web rozdělený na několik oblastí podle zaměření. Pracujeme s daty z následujících dvou oblastí:

- *Hlídač smluv* je vývojově nejstarší součástí webu. Jeho úkolem je zobrazovat smlouvy z registru smluv a umožnit vyhledávání smluv podle kritérií. Oproti webu Ministerstva vnitra nabízí Hlídač smluv několik vylepšení. Veškeré přílohy smlouvy jsou převedeny na prostý text, aby bylo možné vyhledávat i v něm; strojově nečitelné fotografie jsou transformovány na prostý text pomocí OCR. U každé smlouvy je zkontrolována její platnost; smlouva může být označena za neplatnou například v případě, že byla do registru zaslána později než 30 dní od jejího uzavření. U smlouvy se mohou zobrazovat upozornění na její potenciálně problematická místa, například chybějící (skrytou) cenu.
- *Hlídač veřejných zakázek* slučuje data ze všech portálů, kde se nachází informace o veřejných zakázkách (Věstník veřejných zakázek, profily zadavatelů). Vzhledem k povinnému uvádění CPV kódů je snadné si zobrazit zakázky jenom pro určité kategorie. Pro uzavřenou zakázku je uveden výherce tendru. I v této oblasti jsou nečitelné soubory převedeny na text pomocí OCR.

Krátce také zmíníme další oblasti, s nimiž jsou předchozí oddíly propojené:

- *Hlídač osob* obsahuje záznamy politicky aktivních osob; příbuzný *Hlídač sponzorů* obsahuje sponzory politických stran. Pro každou osobu je možné



Obrázek 1.4: Záznam v Hlídači osob a odkazy do dalších částí webu

si zobrazit firmy a subjekty, s nimiž je osoba spřízněná (může se vyskytovat třeba v zastupitelstvu), na jakých smlouvách v registru se osoba podílela, nebo, v neposlední řadě, ve kterých veřejných zakázkách se zobrazují jako dodavatelé nebo zadavatelé.

- Ačkoliv to není oficiální sekcí, součástí webu je databáze subjektů – jak firem, tak úřadů. Pro každý subjekt je možné si zobrazit údaje a statistiky z Hlídače smluv a veřejných zakázek, dále také osoby spojené s firmou nebo úřadem nebo insolvenční záznamy.

- *Hlídač politických financí* stahuje data z transparentních účtu politických stran.

- *Hlídač webů* je na rozdíl od předchozích oblastí techničtěji zaměřený. Jeho úkolem je pravidelně kontrolovat funkčnost webových stránek státních institucí – zda je web přístupný a pokud ano, s jakou odezvou. O funkčnosti webu je následně zobrazena statistika. Zároveň Hlídač webů hodnotí kvalitu zavedení zabezpečeného protokolu HTTPS.

Propojení jednotlivých oblastí a údajů je zásadní pro fungování webu jako nástroje pro analýzu dat. Obrázek 1.4 ukazuje odkazy na ostatní části webu z Hlídače osob.

Technické řešení webu

Web ke svému provozu obecně používá hlavně technologie frameworku .NET a jiné produkty firmy Microsoft. Základem webového frontendu je framework ASP.NET, jako jazyk se používá C#. Pro ukládání dat z registrů se ve velké míře používá Apache Elasticsearch, protože dobře podporuje fulltextové vyhledávání a agregační dotazy. V omezené míře, třeba pro správu uživatelů, je zapojena databáze MS SQL.

Ačkoliv má Elasticsearch některé vlastnosti NoSQL databází, je považován v první řadě za vyhledávací engine. Je schopný ukládat a indexovat dokumenty

(načtené smlouvy, veřejné zakázky) ve formátu JSON. Engine dokáže fungovat distribuovaně a rozdělovat zátěž. [29]

2. Zadání problému

Poté, co jsme si představili kontext této diplomové práce, můžeme přistoupit k samotnému zadání. Stejně jako většina práce odvedená na portálu Hlídač státu, i samotné zadání práce je myšlenka Michala Bláhy. Cílem je vylepšit možnosti analyzování registru smluv tím, že se jednotlivé smlouvy *přiřadí do kategorií*.

V Hlídači smluv je aktuálně možné prohledávat smlouvy pomocí údajů, které obvykle vycházejí z povinných metadat. Je tedy možné hledat smlouvy určitého dodavatele nebo odběratele (podle jmena nebo identifikačního čísla), podle ceny smlouvy, nebo podle data podepsání. Navázané externí databáze vylepšují možnosti vyhledávání: díky znalosti vztahů mezi jednotlivými firmami je například možné hledat smlouvy určitého holdingu firem. Vyhledávací engine je schopný také prozkoumávat samotné texty příloh.

Každá instituce veřejné správy se věnuje jedné nebo více *oblastem zájmu* jako například stavebnictví, zdravotnictví, doprava a mnoho dalších. Tyto oblasti mají typicky více podoblastí: třeba u stavebnictví může jít o stavební inspekci, stavbu samotnou nebo její dokončování. Stejně tak každá smlouva v registru smluv by měla jít přiřadit do některé z kategorií (oblastí nebo podoblastí). Jednu smlouvu ale můžeme zařadit do více kategorií najednou: je možné, že jediná smlouva se týká jak stavebních prací, tak inspekce na stavbě. Samotné kategorie jsou explicitně určeny v rámci zadání a budou popsány dále v této kapitole.

2.1 Motivace a využití

Jaké jsou možnosti využití informace o kategoriích smlouvy? Důležitou motivací je celkové zpřehlednění registru smluv, obzvláště u institucí, které se ze své podstaty věnují mnoha různým okruhům zájmu. Příkladem může být radnice města nebo městské části, která uzavírá smlouvy týkající se:

- nájmu bytů, které vlastní,
- nájmu obchodních prostor, které vlastní,
- rekonstrukce ulic a obecně městského mobiliáře,
- podpory různých občanských spolků,
- právního poradenství a právního zastoupení

a mnoha dalších oblastí. Můžeme ale nabídnout konkrétnější příklady využití kategorií. V případě konkrétnějších podoblastí, může jít třeba o údržbu silnic, kategorizace umožní porovnávat cenovou hladinu v různých regionech a ukazovat na předražené zakázky. Kategorizací smluv lze odhalit fakt, že zakázky v určité oblasti pro určitou instituci jsou stále zadávány jediné firmě, což ukazuje na možnost korupce. Cílem připravovaného projektu Hlídače státu, *K-indexu*, je posoudit každou státní instituci a uvést odůvodněnou pravděpodobnost toho, že v ní dochází ke korupci. Tento projekt přímo počítá s funkční kategorizací smluv.

2.2 Využití veřejných zakázek

Z metadat smlouvy ani z textů příloh, které máme k dispozici, ovšem není možné jednoduše zjistit, do jaké kategorie nebo množiny kategorií smlouva náleží. Tento fakt, že neexistuje přímočará možnost rozpoznávání, nás vede k tomu, že potřebujeme využít možnosti strojového učení. K exaktnějšímu popisu oboru strojového učení se hodí známá definice od Mitchella již z roku 1997 [45]: *Počítačový program se pomocí zkušenosti E učí řešit třídu problémů T, vzhledem k míře úspěchu P, pokud se s přibývající zkušeností E zlepšuje úspěch při řešení T, měřeno pomocí P.* Naší třídou problémů je rozpoznávání smluv, mírou úspěchu je přesnost programu při rozpoznávání (zatím neurčujeme konkrétní metriky). Musíme ale programu dodat E: trénovací data, pomocí kterých se program bude učit.

Přirozenou myšlenkou je, že budeme potřebovat trénovací sadu smluv, které jsou již otagované, tedy označené správnými kategoriemi. Žádná větší sada již otagovaných smluv ale neexistuje:[1] nabízí se ale využít podobnou datovou sadu, které *je* otagovaná a Hlídač státu s ní také pracuje – veřejné zakázky.

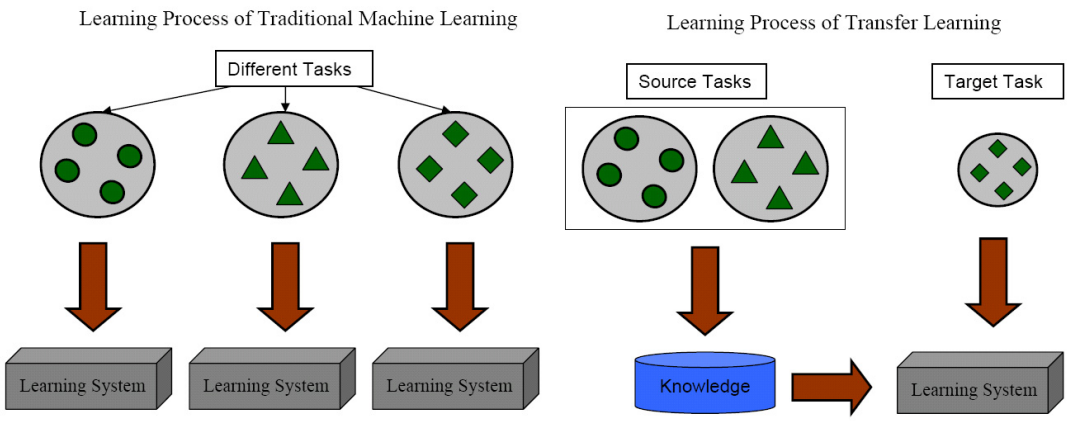
Připomeňme si (viz předešlá kapitola), které údaje máme přístupné u veřejných zakázek:

- název zakázky,
- popis zakázky,
- přílohy (zadávací dokumentace, seznam nutných prací...), převedené do prostého textu,
- údaje o dodavateli, pokud byla zakázka vysoutěžena,
- CPV kód nebo kódy určující oblast veřejné zakázky.

Právě CPV kódy, které mají veřejné instituce povinnost uvádět, nám určí kategorii nebo kategorie, do kterých zakázka náleží. Zakázky a smlouvy jsou si navzájem příbuzné: po vysoutěžení veřejné zakázky je podepsána smlouva s dodavatelem. Alespoň část z textových příloh zakázky se většinou stane i textovou přílohou smlouvy (může jít například o zadávací dokumentaci). Implikace mezi veřejnou zakázkou a smlouvou nicméně platí jen jednostranně – neplatí, že pro jakoukoliv smlouvu lze dohledat příslušnou veřejnou zakázku. Cílem práce je ale právě zjistit, zda tento přístup bude fungovat.

Postup, kdy použijeme trénovací data z jiné domény (datasetu), než pro kterou trénujeme model, se obecně označuje jako *transfer learning* a zjednodušeně je zobrazen na obrázku 2.1. Podrobný průzkum využití tohoto postupu v odborných pracích, který provedli Pan a Yang [52], poukázal na více různých druhů transfer learningu. *Transduktivní učení*, které chceme použít my, se vyznačuje různými doménami trénovací a testovací sady; v obou případech se však řeší stejný problém. Naopak při *induktivním učení* vychází trénovací a testovací sada ze stejné domény, ale liší se problém. Příkladem pro registr smluv by mohlo

[1] Pochopitelně pro následné testování řešení si musíme určitou sadu otagovaných smluv vyrobit. V rámci zadání ale žádný dataset s otagovanými smlouvami přiložen není.



Obrázek 2.1: Ilustrace transfer learningu a porovnání se standardním procesem strojového učení [52]

být předpovídání kategorie na základě cen smluv; veřejné zakázky bychom nevyužívali, stále by ale šlo o druh transfer learningu.

Podrobněji se využitím transfer learningu a jeho alternativami budeme zabývat v sekci 4.5.

2.3 Požadavky na výsledný produkt

Cílem projektu je vyrobit systém, který na svém vstupu přijme smlouvu včetně metadat a na výstup vypíše seznam kategorií, do kterých smlouva pravděpodobně náleží. Ke každé vypsané kategorii může být doplněna pravděpodobnost v procentech. Tato pravděpodobnost se může chápat buď jako *váha* (např. že dokument se z větší části věnuje realitním službám a z menší části stavebnictví), nebo jako *míra jistoty predikce* (např. jsme si jisti, že dokument se věnuje realitním službám, ale je možné, že se také věnuje stavebnictví).

Na portálu Hlídač státu se již nyní používá systém pro klasifikaci smluv, který byl dodán soukromou firmou. Tento klasifikátor však není pro potřeby rozvoje portálu dostatečný a nepredikuje dostatečně přesné výsledky; navíc používá užší sadu kategorií. Jedním z hlavních cílů nového systému je být v predikci lepší, než původní systém.

Technickým detailům se budeme detailněji věnovat až v kapitole 6, týkající se implementace. Prozatím stačí nastínit, že není nutné nový systém těsněji zapojovat do současného projektu. Je dostačující mít k dispozici API, přístupné například přes HTTP (jako REST interface), a veškeré vstupy a výstupy vyřizovat pomocí tohoto API. Z toho důvodu nejsou explicitně dané požadavky na použitý programovací jazyk, framework nebo knihovnu.

Toto řešení odpovídá architektuře *mikroslužeb* (microservices) [4]. Při použití této architektury se rozsáhlejší aplikace skládá z vysoce nezávislých částí, propojených pomocí jasně definovaného API. Při vývoji webového portálu se takto může například oddělit část komunikující s klienty (frontend), část komunikující s databází (backend), nebo správcovský interface. Tím je zajištěna větší technologická nezávislost jednotlivých dílů, snadnější testování a modularita. Pro mikroslužby je typická možnost škálování tím, že se od ní spustí více instancí.

Předpokládá se, že finální klasifikátor se jednorázově spustí na všechny smlouvy v databázi; pravidelně se bude spouštět za účelem klasifikace nově přidaných smluv do registru. Aktuální počet smluv je 476 199 a do registru přibývá průměrně zhruba 2 000 smluv denně. Výsledek klasifikace se uloží do databáze ke smlouvě. Z výkonnostního hlediska tedy není doba klasifikace smlouvy tak výkonnostně kritická, protože nebude probíhat interaktivně, v reálném čase (při načítání webové stránky). Místo toho posuzujeme čas klasifikace celého datasetu smluv; protože tento proces probíhá jednorázově, je v pořádku, aby trval v řádu dní.

Je možné, že klasifikátor bude třeba vylepšovat, buď automaticky, nebo manuálně. V takovém případě bude pravidelně docházet k novému dávkovému zpracování všech smluv a aktualizaci predikovaných kategorií. Dostatečným intervalem může být například jeden měsíc, což stále umožňuje klasifikaci všech smluv v řádu dní.

Zároveň se upřednostňuje, aby výsledný klasifikátor byl *interpretovatelný*, to znamená, že jeho vnitřní struktura bude umožňovat odvodit a pochopit, jak provádí klasifikaci. Důvodem je očekávání ze strany zadavatele, že vzhledem k použití transfer learningu některé z kategorií nemusí fungovat dostatečně dobře, a interpretovatelnost by měla umožnit pochopit tyto nepřesnosti a najít řešení – například úpravou samotného klasifikátoru.

V kapitole 3 zjistíme, že texty některých smluv jsou problematické: jsou zapsané v jiném jazyce (angličtina, čínština), jsou „začerněné“ a tudíž nečitelné, případně se OCR systému nepodařilo z obrázku vyextrahovat text. Počítá se s tím, že tyto texty obvykle vůbec nepůjdou klasifikovat.

Seznam kategorií byl dodán v rámci zadání práce. Pro účely budoucího praktického nasazení výsledného klasifikátoru však není považován za neměnnou věc. Je například možné, že při prohlížení výsledků se některé kategorie ukážou být jako příliš široké a proto je třeba bude podrozdělit.

2.4 Oblasti a kategorie: stručný přehled

Jak již bylo zmíněno, oblasti a kategorie k rozpoznávání byly určeny zadavatelem práce. Jde celkem o 20 oblastí a 104 kategorií. Podrobný seznam je k dispozici v příloze A. Zde uvádíme základní shrnutí. Následující seznam obsahuje všechny oblasti a případně jejich bližší popis:

- Informační technologie
- Stavebnictví
- Doprava
- Strojírenství
- Telekomunikace
- Zdravotnictví
- Potravinářství
- Bezpečnost

  - například armádní, policejní, hasičské vybavení
- Přírodní zdroje
  - například písky, jíly, chemické výrobky
- Energetika
- Zemědělství
- Kancelářské služby
  - například služby tisku, kancelářské stroje, nábytek
- Řemeslná práce
  - například oděvy, textilie
- Sociální služby, vzdělávání a rekreace
- Finance
- Právní a realitní služby
- Technické služby
  - například údržba kanalizace, čistící služby, úklidové služby
- Výzkum
- Marketing
- Ostatní
  - pohostinství
  - služby závodních jídelen
  - průzkum veřejného mínění
  - opravy a údržba – obecná kategorie pro všechny druhy oprav stávajícího zařízení

Každá kategorie používá svůj identifikátor, který je odvozen ze jména oblasti, do níž kategorie náleží, a samotné kategorie. Například `finance-ucetni` označuje kategorii Účetní, revizní a peněžní služby, náležící do oblasti Finance.

S výjimkou oblasti Ostatní obsahuje každá oblast kategorii `*-generic`, která obsahuje veškeré dokumenty z oblasti. Generická kategorie vždy obsahuje minimálně všechny dokumenty náležící do ostatních podoblastí. Někdy ale také obsahuje i jiné dokumenty, které patří do příslušné oblasti, ale nepatří ani do jedné negenerické kategorie.

3. Analýza datových sad

Při využívání metod strojového učení je nezbytnou součástí celého procesu analýza dat. Cílem zkoumání dat je získání přehledu o skutečném obsahu datové sady a kvalitě jednotlivých položek, získání intuice a bližšího porozumění jak datové sadě, tak samotnému zadání. Teprve po analýze je možné uvažovat o dalším postupu, který se typicky skládá z předzpracování dat, využití různých metod strojového učení a porovnání výsledků.

V této kapitole nejprve seznámíme čtenáře s analýzou obou datových sad – veřejných zakázek a smluv. Poté budeme diskutovat výsledky analýzy a rozhodneme, které metody by byly vhodné pro řešení našeho problému.

3.1 Analýza veřejných zakázek

V databázi veřejných zakázek, kterou jsme dostali k dispozici v listopadu 2019, se nachází 429 249 záznamů, avšak jen 140 543 záznamů má uvedený alespoň jeden CPV kód. Protože databázi zakázek budeme využívat hlavně pro trénování na základě CPV kódů, budeme dále uvažovat pouze záznamy s alespoň jedním kódem.

Průměrná velikost textu zakázky (bez započtení dalších metadat) je cca 38 kB, avšak tato hodnota je nevyrovnaná a má poměrně velkou standardní odchylku 222 kB. V zakázkách se průměrně vyskytuje 2,76 příloh, ale i tato hodnota má velkou standardní odchylku 13,6. Zakázky se v registru vyskytují od listopadu 2018. Údaje o datu zveřejnění ale nebudeme dále analyzovat, neboť je nepovažujeme za důležitá pro klasifikaci do kategorií.

Průměrně je k zakázce zadaných 1,7 CPV kódů s odchylkou 1,1. Na základě těchto kódů jsou veřejné zakázky přiřazeny do kategorií, popsaných v sekci 2.4. Rozdělení počtu kategorií je uvedeno v grafu 3.1. Průměrný počet přiřazených kategorií (bez ohledu na to, zda jde o generickou kategorii nebo ne) je 2,87. Za povšimnutí stojí zakázky bez přiřazených kategorií. Těch je pouze 131, z čehož můžeme usuzovat, že předaný seznam oblastí kategorií dobře reprezentuje databázi veřejných zakázek. Relativně malé je množství zakázek s jedinou kategorií (4 284). Zdaleka nejvíce jsou zakázkám přiřazeny dvě nebo tři kategorie (78 245 a 35 124).

3.1.1 Počty zakázek podle oblastí zájmu

Tabulka 3.1 uvádí počty zakázek pro jednotlivé oblasti. Tento počet je vyjádřen pomocí kategorií `*-generic`, které obsahují všechny dokumenty pro danou oblast. Kategorie `jine-*` prozatím vynecháváme, protože ty nemá smysl sdružovat do jedné oblasti. Z tabulky je snadno viditelné, že počty zakázek pro jednotlivé oblasti jsou velmi nevyrovnané. Více než třetina uvažovaných zakázek se týká stavebnictví, silně zastoupené je IT, doprava a strojírenství. Nejméně jsou naopak zastoupené právní služby a potravinářství, které mají méně než tisíc záznamů.

Jak plyne z kapitoly 2, oblasti zájmu nejsou disjunktní; jedna zakázka může náležet do více oblastí najednou.

Kategorie	Počet
agro-generic	8261
bezpecnost-generic	1150
doprava-generic	10801
energie-generic	4079
finance-generic	2474
it-generic	15959
jidlo-generic	669
kancelar-generic	10651
legal-generic	704
marketing-generic	1282
prirodnizdroj-generic	1190
remeslo-generic	1711
social-generic	3935
stav-generic	58528
stroje-generic	18508
techsluzby-generic	5352
telco-generic	6115
vyzkum-generic	1045
zdrav-generic	8931

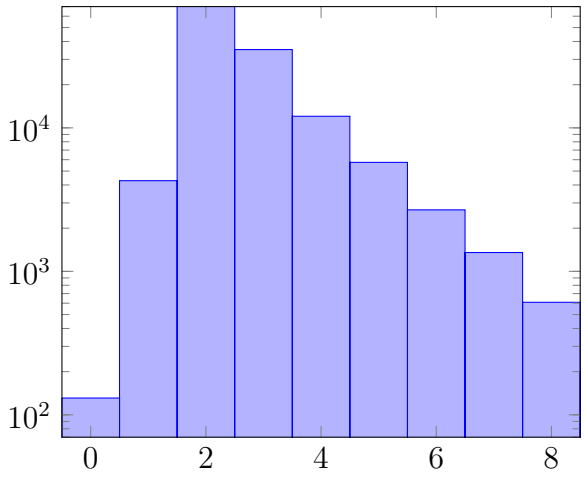
Tabulka 3.1: Počty zakázek pro jednotlivé oblasti zájmu

Prozkoumali jsme počty dokumentů pro průniky oblastí zájmu a průniky kategorií v rámci jednotlivých oblastí. Výsledky jsou znázorněny v příloze B.

U mnoha oblastí zájmu je průnik přirozený, například IT a telekomunikační služby. Některé kategorie jsou problematické z hlediska malého počtu dokumentů, jedná se například o kategorii `stav-voda`, která má pouhých 42 veřejných zakázek. Kategoriím jako `social-zdravotni` nebo `social-vzdelavani` odpovídá jen něco málo přes sto zakázek. Tabulky průniků zvýrazňují fakt patrný již ze zadání kategorií a jejich CPV kódů, totiž že některé kategorie jsou podmnožinami jiných kategorií. Nejedná se přitom jen o kategorie `*-generic` sdružující celé oblasti. Například kategorie `doprava-special` (specializovaná vozidla pro přepravu osob, typicky pohotovostní vozidla) je podmnožinou kategorie `doprava-osobni` (vozidla pro přepravu osob).

3.1.2 Kvalitativní analýza textu

Texty veřejných zakázek byly, na rozdíl od kategorií, zkoumány pomocí metod kvalitativní analýzy, kdy jsme zkoumali náhodně vybrané množiny dokumentů. Třebaže i v tomto případě by bylo možné vytvořit konkrétnější statistiku, její zpracování by bylo o mnoho obtížnější než u kategorií; zatímco kategorií je konečný počet, zde máme k dispozici velké množství různých slov, která, alespoň zatím, nejsou nijak zpracovaná. Cílem této analýzy je tedy také určit, jakým způsobem by texty měly být předzpracovány před použitím metod strojového učení.



Obrázek 3.1: Rozdělení veřejných zakázek podle počtu přiřazených kategorií

Následuje popis různých obecných vlastností textů extrahovaných z databáze veřejných zakázek.

Chybějící popisy a přílohy

Registr smluv vyžaduje, aby uložené dokumenty měly určitou pevnou strukturu. Ve veřejných zakázkách však, na rozdíl od smluv, je mnoho položek ve struktuře nepovinných; to se netýká pouze textu příloh, ale také popisu zakázky. Je možné nalézt zakázky, kterým chybí popis [65], případně nemají k dispozici textovou přílohu [41].

Zvláště u zakázky *Oblek pto potápěče* (sic) [41] je potom znát „minimalistický přístup“ k zadávání údajů, kdy popis prakticky přesně odpovídá názvu zakázky a textové přílohy s dalšími informacemi úplně chybí.

Převaha administrativního stylu

V prakticky všech dokumentech se používá administrativní styl (také úřední nebo jednací styl), zaměřený na fakta a jejich co nejstručnější předání druhé straně. Důležitým znakem je jednotná forma a struktura dokumentů. Časté jsou fráze a klišé. Z našeho hlediska nás zajímá, které části a vlastnosti textu by se daly použít pro úspěšné klasifikování. Mnoho slov používaných v administrativním stylu je totiž „výplňových“, používají se prakticky ve všech dokumentech a nevedou k nějaké bližší představě o kategorii smlouvy. Například následující dokument, týkající se stavby lávky pro pěší a cyklisty v Táboře: [46]

> *Příloha č. 3 zadávací dokumentace*
>
> *Čestné prohlášení o splnění základní způsobilosti dle § 74 odst. 1, 2 a 3 zákona č. 134/2016 Sb., o zadávání veřejných zakázek (dále jen „zákon“) v rámci veřejné zakázky s názvem „Lávka pro pěší a cyklisty a chodníky v prostoru Černých mostů v Táboře“ dodavatele*
>
> *Já, níže podepsaný, jako oprávněný zástupce dodavatele, prohlašuji, že dodavatel splňuje základní způsobilost...*

Nebýt samotného názvu veřejné zakázky, uvedeného na začátku prohlášení, neměli bychom, ani při ručním přiřazování, možnost správnou kategorii zjistit.

Slova a sousloví určují kategorii

Ačkoliv máme mnoho slov, která nejspíše nebudou použitelná pro rozdělení do kategorií, jiné sady slov jsou naopak samy o sobě dostačující k tomu, aby určily kategorii.

Nejlepším příkladem je kategorie `zdrav-leciva`. Veřejné zakázky, týkající se této kategorie, téměř vždy obsahují jméno léku nebo účinné látky, o níž má nemocnice nebo lékař zájem. Například zakázka týkající se dodávky účinné látky Nivolumab [31] nebo cytostatik [23]. Pomocí fulltextového vyhledávání ve veřejných zakázkách je možné si ověřit, že všechny nalezené zakázky opravdu zapadají do kategorie `zdrav-leciva`. Situace se navíc zjednodušuje tím, že u mnoha kategorií mnoho nezáleží na kontextu – například u léčiv není důležité, zda předmětem zakázky je jejich nákup nebo ekologická likvidace.

Takto přímý přístup pochopitelně není možné použít všude. Například u zakázky „Lávka pro pěší a cyklisty a chodníky v prostoru Černých mostů v Táboře“ [46] nelze podle nadpisu jen tak jednoznačně usoudit, že patří do kategorie `stav-mosty`. Zakázka by se například ve skutečnosti mohla týkat instalace kamerového systému.

Podstatná je důležitost samotných slov a jejich významů, bez ohledu na jejich kontext a lingvistické vlastnosti (slovní druh, morfologické kategorie). To si můžeme ilustrovat na armádní zakázce *Run-Flaty pro vojenská vozidla* [42]. Pokud by zakázka neměla žádné další textové přílohy ani popis a nadpis by byl ještě stručnější, například *Nákup Run-Flatů*, znalost významu slova *Run-Flat* by bylo naprosto klíčovým k rozpoznání správné kategorie.[1]

Chybné přiřazení CPV kódů

V sekci 1.3.2 jsme pojednávali o hierarchii CPV kódů a o omezeních tohoto systému. Je obtížné zajistit, aby zaměstanec státní organizace, zodpovědný za vytvoření veřejné zakázky, jí přiřadil odpovídající CPV kódy.

V některých případech chybí správný kód: příkladem může být *Rekonstrukce gastronomického provozu kuchyně v Italské budově* [65]. Zde je správně uveden kód *39314000-6 Průmyslové kuchyňské zařízení*, ale chybí označení faktu, že se jedná o rekonstrukci. Chybně uvedený CPV kód nalezneme například u zakázky *Revitalizace zeleně Halenkov* [49]. Uvedený kód *14212410-7 Ornice* sice na první pohled dává smysl, ale při kontrole nadřazené kategorie se ukáže, že nešlo o správný výběr. Oddíl 14 totiž odpovídá *Produktům těžebního průmyslu, kovovým surovinám a souvisejícím produktům*: užívá se při přímém zpracování těchto látek. Ve skutečnosti ale předmětem zakázky jsou samotné úpravy zeleně, lepším kódem by proto byl některý z hierarchie oddílu 77, *Zemědělské, lesnické, zahradnické služby.*

Funkce jednotlivých příloh

Pokud se veřejná zakázka skládá z více textových příloh, často tyto přílohy mají nevyrovnanou informační hodnotu pro rozpoznání kategorie. Dokumenty hovořící o konkrétní zakázce (typicky zadávací dokumentace) obsahují mnoho

[1] *Run-Flaty* jsou jen jiný název pro dojezdové pneumatiky. Mají speciální konstrukci, která umožňuje dokončit jízdu i v případě defektu.

textu, díky němuž je možné kategorii rozpoznat. Jiné přílohy jsou však obecnější: jako příklad opět může posloužit *Čestné prohlášení o splnění základní způsobilosti*, citované v předešlé části textu. Proto je jistě třeba, aby se zakázka i s přílohami při trénování modelů uvažovala jako celek.

3.2 Analýza smluv

V databázi registru smluv, kterou jsme dostali k dispozici v listopadu 2019, se nachází 476 199 dokumentů. Denně do registru přibývá v průměru 2 000 smluv. Na rozdíl od veřejných zakázek je zkoumání databáze smluv komplikovanější tím, že smlouvy nejsou označené správnými kategoriemi. Chybějící označení nám však nebrání v tom, abychom použili metody strojového učení *bez učitele (unsupervised)*, mezi nimiž je zdaleka nejpoužívanější rozdělení dokumentů do skupin (*cluster analysis*) [21] Následně, stejně jako u veřejných zakázek, provedeme kvalitativní analýzu textu smluv.

3.2.1 Rozdělení smluv do skupin

Cílem rozdělení smluv do skupin je zdůraznit shluky dokumentů, které si jsou nějakým způsobem podobné. Cílem analýzy těchto skupin je identifikovat určité typy smluv, které se v datové sadě vyskytují ve větším množství. Zároveň pro účely dalšího zkoumání je lepší vycházet z nějakého, byť nedokonalého, rozdělení smluv.

Aby bylo možné skupiny posoudit ručně, použili jsme jen velmi malou část datové sady: náhodně jsme vybrali 1000 smluv a nechali jsme je rozdělit do 20 skupin. Použili jsme několik různých metod pro clustering:

- centroidovou metodu K-means,
- hierarchickou aglomerační (bottom-up) metodu,
- metodu měření hustoty vektorového prostoru DBScan,
- hierarchickou metodu BIRCH.

Následující pozorování se souhrnně týká výsledků všech metod; pro účely vyhledávání podobných smluv můžeme konstatovat, že metody vrátily podobné výsledky.

Obsahem několika skupin byly výhradně smlouvy týkající se nákupu léčiv. Potvrdilo se pozorování u veřejných zakázek, že kategorie `zdrav-leciva` je velmi snadno identifikovatelná a oddělitelná od ostatních kategorií. Z 1 000 vybraných smluv se asi 200 z nich týkalo obchodování s léčivy nebo se zdravotnickým materiálem. Zastoupení těchto kategorií v oblasti smluv je tedy znatelně vyšší než u veřejných zakázek.

Do samostatných skupin bylo také zařazených mnoho smluv týkajících se dotací a dotačních programů. Toto zařazení není překvapivé, pokud uvážíme texty těchto smluv. Bez ohledu na skutečnou kategorii se ve smlouvě nachází velké množství ustálených frází. Například u *Dohody o poskytnutí dotace z Programu rozvoje venkova ČR*: [62]

Poskytovatel dotace se za podmínek uvedených v tomto článku zavazuje přiznat a poskytnout příjemci dotace podporu formou dotace na výše uvedenou žádost po splnění podmínek této Dohody a podmínek stanovených v Pravidlech.

Konečná výše dotace se určí na základě skutečně vynaložených, odůvodněných a řádně prokázaných výdajů, na které může být poskytnuta dotace, avšak celková výše přiznané dotace uvedená výše nesmí být překročena.

Podobnost je znát například ve srovnání se smlouvou *Poskytnutí dotace v rámci GP Rozvoj podnikatelů*: [32]

Uznatelné náklady na realizaci akce vznikají nejdříve dnem podpisu této smlouvy oprávněnými zástupci obou smluvních stran.

Čerpáním dotace se pro účely této smlouvy rozumí úhrada celkových nákladů souvisejících s realizací akce, které nejsou touto smlouvou označeny jako náklady neuznatelné. Celkové náklady akce ve skutečné výši musí být vyúčtovány, uhrazeny a promítnuty v účetnictví Příjemce nejpozději do dne uvedeného v ČI. 7 odst. 1 této smlouvy.

Řazení dotací do stejné skupiny však z hlediska rozpoznávání kategorií není příliš nápomocné, protože dotační programy se týkají mnoha různých kategorií.

3.2.2 Kvalitativní analýza textu

Podobně jako u veřejných zakázek, i zde jsme provedli kvalitativní analýzu dostupných textů. Obecně se dá říci, že vlastnosti textů jsou podobné. Jedním z důvodů je fakt, že uzavření smlouvy často (i když ne vždy) navazuje na vyřízení veřejné zakázky; některé texty z veřejných zakázek, typicky zadávací dokumentace, se poté vloží i do textu smlouvy. Mezi podobné vlastnosti, o nichž jsme již pojednávali v části 3.1.2, patří:

- převaha administrativního stylu,
- funkce jednotlivých příloh (samostatné přílohy často nejsou dostačující k určení kategorie),
- slova a sousloví určují kategorii.

Nyní pojednáme o vlastnostech, které jsou specifické pro texty smluv.

Nezpracovatelný text příloh

V sekci 1.3.1 jsme zmiňovali, že v registru smluv je povinné zveřejňovat textové přílohy smluv ve strojově čitelném formátu. V praxi se ukázalo, že úřady někdy tento proces obcházejí tím, že do příloh s povoleným typem souboru (například PDF) ukládají fotografie textu smlouvy. Příkladem může být *Dodatek č. 7 ke Smlouvě o umístění technického zařízení na sběr* [47], kde PDF soubory jsou tvořené pouze fotografiemi. Ačkoliv portál při zpracování nových smluv převádí obrázky pomocí OCR na text, ne vždy je tento převod dokonalý.

Jiným problém jsou cizojazyčné smlouvy, nejčastěji v anglickém jazyce, například *Contract on the use of the high performance computing cluster* [64]; vyskytují se také smlouvy v čínštině.

Jak již bylo rovněž zmíněno v sekci 1.3.1, problematickým bodem registru je otázka obchodního tajemství. Ve zveřejněném textu smlouvy je povoleno vynechat („začernit“) osobní údaje nebo jiné části za účelem nevyzrazení obchodního tajemství. Text některých smluv je ale díky této možnosti zakryt téměř celý. Nejlepším příkladem jsou prakticky všechny smlouvy od Budějovického Budvaru, uvést můžeme třeba *Tiskovou produkci billboardů* [14], kde je zakrytý takřka všechen text, cena smlouvy a dokonce i identita protistrany.

Je zajímavé, že přílohy smluv od některých úřadů využívají formátu XML a mají dokonce specifikované schéma. Jako příklad si můžeme uvést *Velký Beranov 356 - RD* [43]. Můžeme si všimnout, že z XML přílohy je extrahovaný textový obsah, tedy všechny textové řetězce, které nejsou součástí XML tagů nebo komentářů.

3.3 Shrnutí

Krátce si zopakujme požadavky, které jsme již uváděli v kapitole 2. Konkrétní technické řešení může být nezávislé na infrastruktuře webu Hlídač státu, pouze musí být k dispozici API. Časová výkonnost klasifikátoru není kritická, klasifikace může probíhat dávkově (např. jednou za měsíc). Protože smlouvy ke klasifikaci nejsou vždy korektní (mohou chybět důležité údaje), je v pořádku, pokud některé z nich nebude možné správně klasifikovat. Seznam kategorií, do nichž máme smlouvy rozdělit, není neměnný: možná bude třeba některé kategorie podrozdělit, případně sloučit.

V této kapitole jsme analyzovali jak smlouvy, tak veřejné zakázky. Dokumenty obsahují velké množství nedůležitých, „výplňových“ slov, což je typická vlastnost administrativního stylu. Může se stát, že správná kategorie smlouvy je určena malým množstvím slov. CPV kódy ve veřejných zakázkách jsou někdy přiřazeny nesprávně.

Procentuální zastoupení kategorií ve smlouvách se nedá přesně určit. Díky rozložení podmnožiny smluv na skupiny jsme však zjistili, že se liší od procentuálního zastoupení kategorií ve veřejných zakázkách.

Problém, který řešíme, má některé specifické vlastnosti, na které musíme brát zvláštní ohled:

- trénovací a testovací sady pochází z různých domén,
- trénovací a testovací sada mají různé rozložení četností kategorií,
- jedna smlouva může být zařazená do většího množství kategorií,
- všechny kategorie nejsou na stejné úrovni: existují kategorie, které jsou nadmnožinami jiných kategorií.

Se všemi požadavky a výsledky analýzy budeme pracovat v další kapitole, kde si představíme možná řešení klasifikace smluv.

4. Popis řešení

V této kapitole si představíme řešení problému klasifikace smluv. Nejprve určíme celkový postup a jednotlivé kroky (předzpracování, extrakce příznaků apod.), následně budeme diskutovat o metodách, které použijeme pro implementaci každého z kroků.

V jednotlivých sekcích také vybíráme konkrétní implementaci zvoleného řešení. Výběr je ovlivněn tím, že jako základní prostředí jsme si zvolili jazyk Python. Diskuze k tomuto rozhodnutí je v sekci C.1.

4.1 Základní terminologie

V této sekci shrneme základní principy strojového učení a zároveň uvedeme termíny, které budeme dále používat.

Budeme pracovat s *datovými sadami*, které se skládají z *dokumentů*. Každý dokument může být zařazený do určitého množství *kategorií*; je ale možné, že správné zařazení není známé.

Metody strojového učení lze rozdělit do několika druhů.

Při *učení s učitelem (supervised learning)* máme k dispozici *trénovací* a *testovací* datovou sadu, přičemž všechny dokumenty v obou sadách jsou zařazené do kategorií. *Klasifikátor* je funkce, která na vstupu přijme dokument a na výstupu vrátí *predikci* – množinu kategorií, do nichž, podle klasifikátoru, dokument náleží.

Při *učení bez učitele (unsupervised learning)* máme k dispozici datovou sadu, avšak dokumenty v této sadě nejsou zařazeny do kategorií. Dokumenty můžeme například rozdělovat do skupin podle podobnosti.

Další druhy (například *reinforcement learning*) nebudeme uvádět, protože se používají k řešení jiných problémů.

4.2 Celkový postup řešení

Učení bez učitele jsme již využili v kapitole 3 pro zkoumání struktury a vlastností datové sady. Tento druh učení však neumožňuje sestavit klasifikátor pro určení kategorií, protože nemá k dispozici informaci o korektním zařazení dokumentů do kategorií. Musíme tedy využít učení s učitelem.

Celkový postup při učení s učitelem se skládá z několika kroků, které je možné nalézt ve většině zdrojů, s nimiž jsme pracovali [37; 30; 20]. Následuje přehled kroků:

- *předzpracování (preprocessing)* zdrojových dokumentů v použitých datových sadách: extrahujeme textový obsah, normalizujeme textový obsah a převedeme do správného kódování a podobně,
- *extrakce příznaků (feature extraction)*: z obsahu každého dokumentu vygenerujeme vektor příznaků, který bude v následujících krocích dokument reprezentovat,
- *snižování dimenzionality*: kvůli výhodnější časové a paměťové efektivitě je možné snížit dimenzi vektorů příznaků,

- *učení modelu*: vybraná metoda klasifikace přijme zpracovanou trénovací sadu a vytvoří klasifikátor,
- *testování modelu*: necháme klasifikátor predikovat kategorie pro dokumenty v testovací datové sadě, porovnáme tuto predikci se skutečným ohodnocením dokumentů a určíme výkonnost klasifikátoru.

V následující sekci si představíme postup při předzpracování textu. Poté se rozhodneme, které metody klasifikace použijeme – až na základě vybraných metod totiž budeme rozhodovat o postupu při extrakci příznaků a snižování dimenzionality.

Testováním modelu se budeme zabývat až v příští kapitole, týkající se experimentů.

4.3 Předzpracování textu

V této sekci rozhodneme o tom, které textové části smluv a veřejných zakázek budou využity pro klasifikaci. Také vysvětlíme, jaké úpravy textu budou provedeny předtím, než text použijeme v dalších krocích.

4.3.1 Textový obsah

Textový obsah smlouvy, jejíž strukturu jsme popsali v sekci 1.3.1, se skládá z následujících částí:

- jména a adresy objednatele (státního subjektu),
- jména a adresy ostatních smluvních stran,
- předmětu smlouvy,
- textového obsahu všech příloh.

Textový obsah veřejné zakázky, jejíž struktura byla popsána v sekci 1.3.2, je podobný. Skládá se z následujících částí:

- jména objednatele (státního subjektu),
- jména dodavatele nebo dodavatelů, pokud jsou specifikováni,
- názvu a popisu veřejné zakázky,
- textového obsahu všech příloh.

Pro účely klasifikace použijeme všechny části a budeme je uvažovat jako celek. Alternativou by bylo reprezentovat dokument jako soubor čtyř textových částí. Pak by však nebylo jasné, jak bychom měli tyto části zkombinovat při trénování klasifikátorů; žádný zdroj, ze kterých jsme čerpali, podobnou variantu nevyužíval.

Další možností by bylo vyřadit některé části, tím se však připravujeme o cenná data. Určitě se nechceme zbavit příloh a popisů, protože tyto části jsou průměrně nejdelší a nejpodrobnější; zároveň ale máme zájem i o jména a adresy firem, protože i tyto údaje mohou o kategorii rozhodnout (velké množství firem se věnuje jednomu vyhraněnému oboru).

4.3.2 Úpravy textu

Získaný text každé smlouvy nebo zakázky je třeba pročistit a zbavit nepotřebných informací, aby byl pro klasifikaci co nejpoužitelnější. Veškeré použité úpravy vycházejí z článku Kowsari a kol. [30]. V jiných zdrojích je však postup podobný.

Připomínáme, že texty jsou již převedeny ze zdrojových typů soboru (PDF, DOC apod.) do prostého textu.

Normalizace Unicode

Extrahované texty všech dokumentů používají kódování textu UTF-8, které využívá normu kódování Unicode. České znaky ale mohou být v Unicode kódovány více způsoby: například znak `Á` je uveden v Unicode pod kódem `U+0041`, lze však také zapsat jako kombinaci znaků `U+0041` (obyčejné `A`) a `U+0301` (čárka, `COMBINING ACUTE ACCENT`). Text v Unicode tedy musíme normalizovat, aby bylo zajištěné, že každý znak bude reprezentován jediným možným způsobem. Existují čtyři normalizační formy [16].

Normy se odlišují v první řadě tím, které znaky považují za identické. V případě *kanonické ekvivalence* se za identické považují fakticky stejné znaky, které jsou jen různě zapsané (viz příklad s `Á`); v případě *kompatibilní ekvivalence* se za identické považují takové znaky, které lze *ve specifických případech* považovat za identické (příkladem mohou být různé ligatury).

V druhé řadě se normy odlišují tím, zda znaky ve výsledku ponechávají rozložené, nebo zda je skládají dohromady. Kombinace těchto dvou rozhodovacích faktorů nám dá čtyři různé formy.

Rozhodli jsme se použít normu NFKC, *Normalization Form Compatibility Composition*, která znaky skládá a považuje je za identické na základě kompatibilní ekvivalence. Rozhodli jsme se pro skládání, protože alespoň v unixových systémech je vnímáno jako standardnější. Pokud jde o ekvivalenci znaků, alespoň pro české znaky použitá ekvivalence nehraje roli. V každém případě by bylo možné využít kteroukoliv z norem, jde pouze o to, aby celý dataset byl převeden do jediné z nich.

Tokenizace

Tokenizace označuje postup, kdy jsou z textu extrahována jednotlivá slova. Tato úprava je nutná, protože při extrakci příznaků pracujeme právě s jednotlivými slovy.

Normalizace tvarů slov

Normalizací tvarů chceme docílit toho, aby různé lingvistické tvary téhož slova, například různé skloněné podstatné jméno (*kočka*, *koček*...) byly posuzovány jako jediné slovo. U podstatných jmen jde obvykle o převod do jednotného čísla a prvního pádu; u sloves do infinitivu.

Je převod slov skutečně nutný? Manning a kol. [37] se zmiňují o tom, že například při zpracování anglického textu převod slov na základní tvar příliš

nezvyšuje úspěšnost metod. Důležitou funkci má tato fáze úpravy textu při zpracování flexivních jazyků, tedy těch jazyků, kde gramatické funkce slov jsou vyjadřovány pomocí *flexe (ohýbání)* slova. Mezi typické zástupce flexivních jazyků patří slovanské jazyky, mezi nimi i čeština. Jiná situace je u angličtiny, která se počítá mezi *analytické* jazyky – zde je využívání flexe minimální, namísto toho se pracuje například se slovosledem.

Existují dva druhy normalizace tvarů slova. Cílem *stematizace* je najít kmen slova, tedy jeho část, která se během ohýbání nemění. Slovo *přijdu* se tedy při stematizaci změní na *přij*. *Lemmatizace* naproti tomu hledá *lemma*, základní tvar slova – například u podstatných jmen jde o převod na slovo v prvním pádu jednotného čísla, v případě sloves na infinitiv a tak podobně. Slovo *přijdu* se při lemmatizaci změní na *přijít*.

Měli bychom využít stematizaci nebo lemmatizaci? Na rozdíl od lemmatizace, použití stematizace by pro nás mělo dvě nevýhody: za prvé, výsledek stematizace různých slov může vrátit stejný výsledek (například slova *led* a *leden*); za druhé, při použití metod klasifikace, kde pracujeme s jednotlivými slovy, bude analýza a práce s lemmatizovanými slovy snazší než se stematizovanými slovy. Pro lemmatizaci českého textu navíc existují dostupné nástroje (viz popis implementace), proto jsme se rozhodli využít lemmatizaci.

Odstranění nedůležitých slov

V této fázi se chceme zbavit slov, která určitě nepomohou ke správnému určení kategorie slova. Touto fází snížíme množství slov, které se dostane do dalších fází, což pomáhá z výkonnostního hlediska, ale také by mělo zlepšovat výsledky klasifikace. Mnoho metod klasifikace funguje lépe, pokud na vstupu dostane menší množství důležitějších slov. Na druhou stranu toto zlepšení nelze přeceňovat; trendem mezi metodami klasifikace je spíše zařídit, aby použitá nepotřebná slova byla rozpoznána a ignorována automaticky [37]. Z toho důvodu jsme při výběru způsobu rozeznání nedůležitých slov rozhodovali podle toho, jak velké množství slov dokáže odfiltrovat.

Existuje několik způsobů sloužících k vybrání nedůležitých slov, například:

- statický seznam nedůležitých slov – obsahuje obvykle nejčastěji používaná slova v daném jazyce,
- nedůležitá jsou slova, která se vyskytují ve velké části dokumentů v datové sadě (například v 80 % dokumentů),
- nedůležitá jsou slova s určitými vlastnostmi (krátká délka, slovní druh).

Rozhodli jsme pro odstranění slov využít funkci rozpoznání slovních druhů (typicky součást lemmatizéru), protože dokáže odstranit největší množství slov a zároveň máme jistotu, že odstraněná slova nijak nepomohou ke správné klasifikaci. V případě odstranění slov, která se vyskytují ve velké části dokumentů, není úplně jisté, že by tato slova nijak nepomohla. Odstranění krátkých slov také není vhodné, protože dokumenty často obsahují zkratky, které mohou vést ke správnému rozpoznání. Statické seznamy nedůležitých slov, obvykle nejsou dostatečně rozsáhlé, aby stačily samy o sobě.

V dokumentech jsme ponechali podstatná a přídavná jména, slovesa a příslovce. Mezi vyřazenými slovními druhy jsou číslovky: cenové údaje ve smlouvách a zakázkách jsme se rozhodli vynechat, protože z analýzy nevyplývalo, že by byly zvlášť nápomocné pro určení kategorií.

Nebyli jsme si jisti, zda do dalšího zpracování ponechat příslovce; pro češtinu je například Onderka [50] ponechává, ačkoliv jeho cílem však bylo určovat emocionalitu textu.

Vytvoření n-gramů

Některé z metod klasifikace dokáží pracovat se samostatnými slovy, avšak ignorují pořadí slov v dokumentu a kontext jednotlivých slov. Řešením v takovém případě je použít *n-gramy* – meta-slova, která se skládají z n slov, která jsou ve zdrojovém dokumentu vedle sebe. Pokud se kupříkladu v dokumentu vyskytuje slovní spojení *plyšový medvěd*, můžeme z něj extrahovat nejen slova *plyšový* a *medvěd*, ale i 2-gram *plyšový=medvěd*. Pokud využíváme n-gramy o délce n, vkládáme na vstup metody klasifikace i n-gramy o nižších délkách a jednotlivá slova. To znamená, že zvyšováním n nepřicházíme o žádné informace, pouze získáváme nové.

Výhodou použití n-gramů je obvykle vyšší výkonnost metod klasifikace, nevýhodou je velký nárůst množství vstupních slov. Čím větší n může být použito, tím větší tento nárůst je. Výzkum Fürnkranze [19] ukazuje, že největší zlepšení přineslo generování 2-gramů a 3-gramů; další zvětšování n nemělo na výsledky velký efekt a vedlo ke zpomalení běhu.

Abychom zabránili přílišnému nárůstu množství n-gramů, rozhodli jsme se uvažovat pouze ty n-gramy, jejichž slova jsou v originálním textu těsně vedle sebe a nejsou rozdělena žádným vyřazeným slovem. Například ze sousloví *Dlouhý, Široký a Bystrozraký* bychom extrahovali 2-gram *Dlouhý=Široký*, ale ne *Široký =Bystrozraký*.

4.3.3 Implementace úprav textu

Samotné provedení normalizace Unicode je poměrně přímočaré – použili jsme vestavěný modul jazyka Python `unicodedata` a jeho funkci `normalize`.

Složitější rozhodování vyžadovala tokenizace a lemmatizace. Protože se na webovém portálu Hlídače státu se pro uchování dat primárně používá server Apache Elasticsearch, zkoumali jsme jeho vestavěné funkce. Alespoň ve využívané verzi serveru na portálu však server nabízí pouze stematizaci, nikoliv lemmatizaci [5]. Jednoduchým a funkčním řešením se ukázaly být nástroje vyvíjené na Ústavu formální a aplikované lingvistiky MFF UK, zvláště pak UDPipe [60], který dokáže postupně provést tokenizaci, lemmatizaci a detekci slovních druhů. UDPipe pro svoji funkci využívá nástroj pro morfologickou analýzu MorphoDiTa [61]. V roce vydání článku dosahoval jazykový model MorphoDiTy pro češtinu nejlepšího výsledku v lemmatizaci. Novějším příspěvkem z řad autorů je *LemmaTag*, systém pro lemmatizaci založený na rekurentních neuronových sítích [28]. Ačkoliv tento systém dosahuje lepšího výsledku lemmatizace českých slov, na rozdíl od původních nástrojů nenabízí možnost tokenizace a napojení na UDPipe. Proto jsme se rozhodli použít MorphoDiTu jako systém pro tokenizaci a lemmatizaci a UDPipe jako frontend.

MorphoDiTa rovněž zajistila detekci slovních druhů slova, podle nichž jsme provedli filtraci výsledku.

Vytvoření n-gramů z lemmatizovaných slov jsme implementovali sami, protože jde o jednoduchý a přímočarý proces.

4.4 Metody klasifikace

Tato sekce se zabývá konkrétními klasifikační metodami, které budeme používat. Způsob extrakce příznaků a případného snižování dimenzionality je pro každou metodu specifický, a proto je v každé sekci uveden zvlášť. Nejprve však musíme metody klasifikace vybrat a tento výběr odůvodnit.

4.4.1 Výběr metod

Existuje velké množství metod klasifikace, které je možné použít pro řešení našeho problému. Abychom získali dostatečný přehled, vycházeli jsme z rešeršních článků, týkajících se metod klasifikace textu. Konkrétně jde o články Kowsari a kol. z roku 2019 [30] a Patra a kol. z roku 2013 [54].

Agregace binárních klasifikátorů

Nejsilnějším omezením při výběru metod se ukázal být požadavek na to, že jedna smlouva může náležet do většího množství kategorií. Některé klasifikační metody jsou uzpůsobeny pouze pro *binární problémy*, kdy výsledkem klasifikace může být buď hodnota typu true/false, případně hodnota pravděpodobnosti v intervalu [0,1] – typickým příkladem je logistická regrese. [58] Další klasifikační metody lze použít pro řešení *multiclass* problémů, kdy dokument náleží do právě jedné kategorie – příkladem může být SVM. [13] Metody pro *multilabel* problémy, kdy dokument může náležet do více kategorií, jsme nalezli – například FastText [26] – tvoří ale spíše menšinu.

To však neznamená, že není možné využít jiné než multilabel klasifikátory. Lze zkombinovat více binárních klasifikátorů a zajistit, aby jejich agregace predikovala řešení multilabel problému. Existují dvě možné metody agregace; n zde představuje počet kategorií.

- *One-vs-all:* pro každou z kategorií vytvoříme binární klasifikátor. Vstupem každého klasifikátoru je dokument, výstupem je pravděpodobnost, že dokument náleží do příslušné kategorie. Na základě pravděpodobností vrácených těmito n klasifikátory se určí predikovaný seznam kategorií.

- *One-vs-one:* vytvoříme binární klasifikátor pro každou dvojici kategorií; budeme mít tedy $\binom{n}{2}$ klasifikátorů. Každý z klasifikátorů vyjadřuje pravděpodobnost, zda dokument náleží do jedné, nebo druhé kategorie. Na základě výsledků všech klasifikátorů se opět určí predikce.

Pro naše účely je však použití agregace problematické. Počet kategorií, 104, je rozhodně příliš velký, než aby bylo možné využít metodu *one-by-one*: teoreticky bychom museli natrénovat 5 356 klasifikátorů. V praxi by tento počet byl nižší

(některé kategorie jsou podmnožinami jiných), ale i tak stále velmi vysoký. Metoda *one-vs-all* se zdá být použitelnější, avšak počet klasifikátorů k natrénování je stále velký vzhledem k tomu, že v rámci experimentů chceme otestovat různé parametry metod klasifikace. Vzhledem k velikosti datové sady jsme usoudili, že tímto způsobem není možné one-vs-all klasifikátory dostatečně otestovat.

Alternativou by mohlo být natrénování klasifikátorů na menší datové sadě, případně netestování parametrů. To by však znamenalo zásadní rozdíl od testování skutečných multilabel metod, kde můžeme využít celou datovou sadu a parametry ladit budeme. Proto jsme se rozhodli one-vs-all klasifikátory nezařazovat do vybraných metod.

Diskuze nad jednotlivými metodami

Na základě rešeršních článků jsme sestavili seznam metod klasifikace, které autoři článků zkoumali:

- logistická regrese,
- bayesovské klasifikátory,
- Support Vector Machines (SVM),
- rozhodovací stromy,
- náhodné stromy (random forests),
- hluboké učení (deep learning),
- metody založené na word embeddings.

Poznamenejme jen, že definice hlubokého učení není úplně ustálená. Různé zdroje se neshodují v tom, zda word embeddings je jedna z metod hlubokého učení, nebo ne. V námi použitých článcích však byla zařazována odděleně, proto tento přístup používáme i zde.

Ze seznamu jsme vyřadili logistickou regresi, bayesovské klasifikátory a SVM, protože je lze použít pouze pro binární rozhodování a multiclass problémy.

Metody hlubokého učení zaznamenaly v posledních letech velkou popularitu, ale rozhodli jsme se je do této práce nezařadit. Existuje poměrně mnoho různých struktur neuronových sítí, které je v hlubokém učení možné použít (hluboké neuronové sítě, rekurentní sítě apod.) a obvykle je nutné podrobně ladit parametry těchto sítí. Klasifikace textu pomocí těchto sítí by byla tématem na samostatnou práci. Dalším důvodem pro nepoužití metod hlubokého učení je to, že pro naše účely nesplňují nároky na snadnou interpretovatelnost (viz sekce 2.3).

Metoda náhodných stromů je silnější verze rozhodovacích stromů, respektive jde o trénování více rozhodovacích stromů najednou (viz sekce 4.4.3). Proto z těchto dvou metod použijeme právě náhodné stromy.

Rovněž použijeme metody založené na word embeddings, kde využití slovních vektorů (viz sekce 4.4.2) by mělo zajistit dostatečnou interpretovatelnost.

Klíčová slova

Nad rámec metod klasifikace vybraných ze článků chceme otestovat ještě jeden klasifikátor, jenž rozpoznává kategorie na základě *klíčových slov* – takových slov nebo n-gramů, které jsou specifické pro určitou kategorii. Cílem je dosáhnout toho, aby ke každé kategorii existovala relativně malá sada klíčových slov, do které se dá ručně zasahovat a upravovat. V porovnání s ostatními metodami klasifikace je tento způsob nejefektivnější z hlediska interpretovatelnosti; rovněž zajišťuje možnost snadného slučování a rozdělování kategorií v již existujícím modelu.

Shrnutí

Následující část této sekce popíše jednotlivé metody klasifikace, které jsme vybrali. Jde o metody:

- word embeddings,
- náhodné stromy,
- klíčových slov.

4.4.2 Word embeddings a odvozené metody

Obecným cílem metod založených na *slovních vektorech (word embeddings)* je přiřadit každému slovu v databázi bod ve vícedimenziálním prostoru tak, aby významově příbuzná slova byla navzájem umístěna blíže než nepříbuzná slova. Učení slovních vektorů je obvykle založeno na posuzování kontextu, v němž se slovo nachází; kontext je pak jedním ze vstupů neuronové sítě. Obecněji však trénování vektorů nemusí být nutně spojené s neuronovými sítěmi, jak ukazují například Lebret a Collobert [35], kteří využili metodu PCA (viz sekce 4.4.3). Tyto postupy však ve zkoumaných pracích vedou k horšímu výsledku, proto v naší práci budeme využívat neuronové sítě.

Úvod do neuronových sítí

Pro pochopení následujícího textu o slovních vektorech zde uvádíme základní pojmy a postupy související s neuronovými sítěmi, čerpané z práce Reeda a Markse [55].

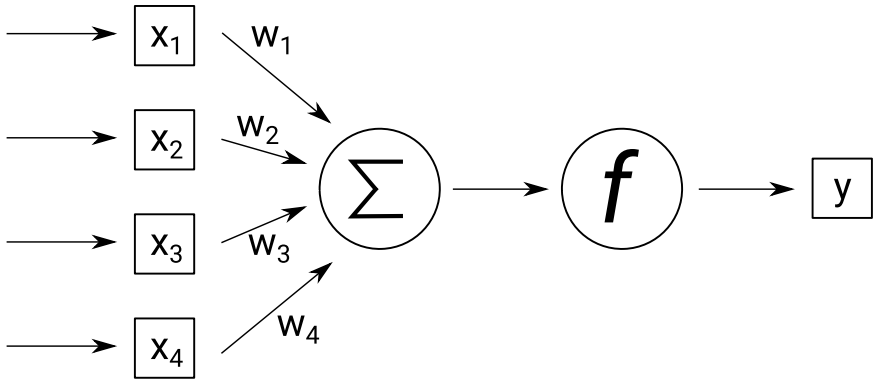
Struktura umělých neuronových sítí, využívaných v informatice, volně vychází z nervové soustavy živočichů. Základní jednotkou v nervové soustavě je *neuron*, buňka, která reaguje na vzruchy přicházející z jiných neuronů skrze výběžky zvané *dendrity*. V závislosti na síle těchto vzruchů buňka může generovat vzruch a předávat jej dalším neuronům přes výběžek zvaný *axon*.

Základní jednotka umělé neuronové sítě se rovněž nazývá neuron. Výstupní hodnota neuronu je určena funkcí:

$$y = f\left(\sum_{k=1}^{N} w_k x_k\right)$$

N je počet neuronů, jejichž výstupní hodnoty současný neuron zpracovává. Hodnota x_k je výstupní hodnota k-tého neuronu; hodnota w_k je váha hodnoty k-tého neuronu. f se nazývá *aktivační funkce*. Tato funkce převede součet všech hodnot, vynásobených vahami, na výstupní hodnotu neuronu.

Schématické znázornění vstupu a výstupu neuronu je na obrázku 4.1.



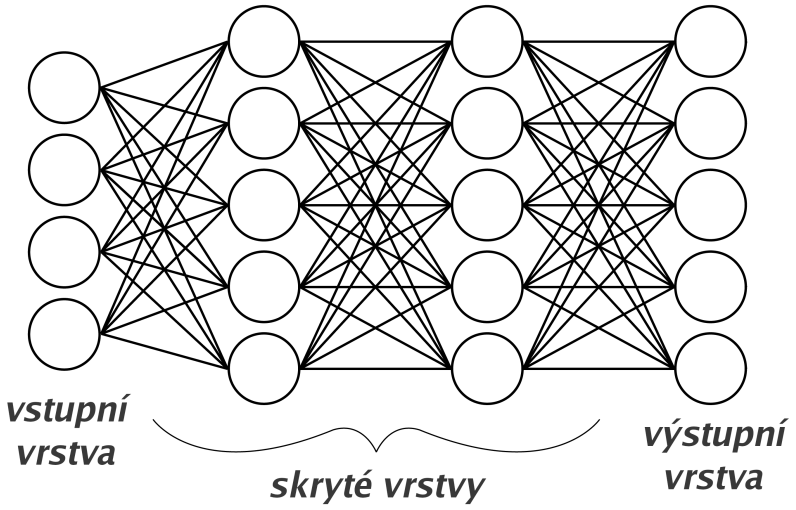
Obrázek 4.1: Schéma vstupů a výstupů a zpracování hodnot pro jeden neuron v neuronové síti

Neuronová síť přijímá vstup jako vektor číselných hodnot, výstupem je opět vektor číselných hodnot. Vstupním číselným vektorem může být například reprezentace určité části textu dokumentu; výstupní hodnoty vyjadřují pravděpodobnost, že dokument náleží do některé z kategorií. Z toho vyplývá, že samotnou neuronovou síť využíváme jako klasifikátor.

Síť se typicky skládá z *vrstev*, kde každá vrstva obsahuje určitý počet neuronů. Neurony ve vrstvě L mohou dostávat vstupy pouze z neuronů z předcházející vrstvy $L-1$, svůj výstup předávají pouze do neuronů na následující vrstvě $L+1$. První vrstva se nazývá *vstupní*, do této vrstvy se vkládá reprezentace vstupní hodnoty; výsledek neuronové sítě se vrací z poslední, *výstupní* vrstvy. Ostatní vrstvy kromě těchto dvou se nazývají *skryté*.

Obrázek 4.2 znázorňuje příklad neuronové sítě. Z nákresu vyplývá, že pro výpočet výstupních hodnot pro určitý vstup je třeba procházet síť po jednotlivých vrstvách. První skrytá vrstva bere vstup ze vstupní vrstvy sítě a vydá svůj výstup. Hodnoty na tomto výstupu slouží jako vstup druhé skryté vrstvy, která na jeho základě vypočte svůj výstup. Tímto postupem nakonec stanovíme výstupní hodnoty celé sítě na výstupní vrstvě.

Aby neuronová síť řešila správně určitý problém, je třeba ji natrénovat: během tréninku upravujeme váhy pro jednotlivé neurony. Pro trénink potřebujeme množinu vstupů a jejich správných (očekávaných) výstupů. Na začátku tréninku inicializujeme váhy všech neuronů na náhodné hodnoty. Pro každou dvojici vstup-výstup provedeme následující: vložíme vstup do vstupní vrstvy a necháme síť vypočítat výstup – a pokud se tento výstup nerovná správnému výstupu, upraví se váhy neuronů takovým způsobem, aby případná další odpověď sítě na vstup více blížila správnému výstupu.



Obrázek 4.2: Příklad neuronové sítě se dvěma skrytými vrstvami. Kolečka znázorňují neurony, čáry vztahy mezi nimi (výstupní hodnota jednoho neuronu je vstupem druhého)

Statistický model jazyka

Jako *statistický model jazyka* označujeme pravděpodobnostní model, zabývající se výskytem slov určitého jazyka ve zkoumaných dokumentech. Model může být reprezentován například jako podmíněná pravděpodobnost výskytu určitého slova na základě výskytu předešlých slov [7]:

$$\hat{P}(w_1^T) = \prod_{t=1}^{T} \hat{P}(w_t|w_1^{t-1}) \tag{4.1}$$

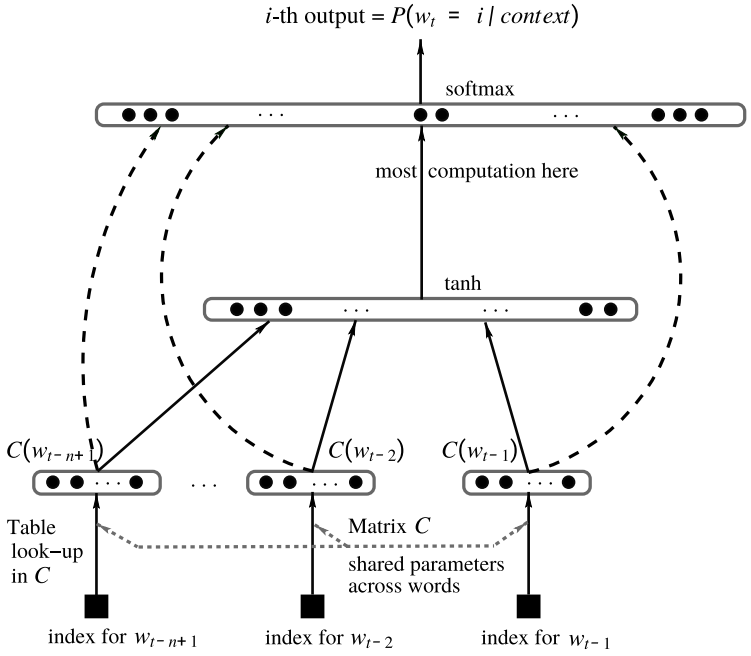
Zde w_t označuje slovo v dokumentu na t-té pozici a w_i^j označuje seřazenou n-tici slov: $w_i^j = (w_i, w_{i+1}, \cdots, w_{j-1}, w_j)$. Obvykle ale předpokládáme, že významnější statistickou závislost mají slova, která jsou blízko u sebe a vzdálenější slova vynecháváme. Ve výpočtu podmíněné pravděpodobnosti výskytu slova proto ponecháváme jenom jeho *kontext*, tedy $n-1$ posledních slov:

$$\hat{P}(w_t|w_1^{t-1}) \approx \hat{P}(w_t|w_{t-n+1}^{t-1}) \tag{4.2}$$

Tento model označujeme jako *n-gramový model jazyka*. Problém však představuje praktické modelování těchto kontextů, které může představovat paměťovou i časovou zátěž. Platí to zvláště v případě, kdy zvětšujeme hodnotu n. Mějme databázi 100 000 slov; pokud bychom chtěli analyzovat skupiny slov po deseti, která se v nějakém textu vyskytují vedle sebe, pracovali bychom potenciálně až se $100\,000^{10}$ záznamy, což není únosné. I v případě, že by informace o kontextech byly zpracovatelné, stále bychom čelili problému vysoké dimenzionality problému, obvykle uváděnému pod pojmem *curse of dimensionality* [33].

Statistický model NNLM

Bengio a kol. [7] představili model obvykle nazývaný jako NNLM (*Neural network language model*). Tento model řeší problém vysoké dimenzionality právě pomocí slovních vektorů: místo konkrétních slov pracujeme s vektory o dimenzi, která je řádově nižší než počet slov; obvykle jde o vyšší desítky nebo nižší stovky dimenzí. Každému slovu v dokumentu je přiřazen vektor; podobné vektory budou mít ta slova, která mají podobnou sémantickou a syntaktickou funkci ve větách.



Obrázek 4.3: Struktura neuronové sítě navržená Bengio a kol. [7]

Trénování vektorů probíhá pomocí neuronové sítě, jejíž struktura je vyobrazena na obrázku 4.3. Mějme slovník o $|w|$ slovech a dokumenty, v nichž jsou slova použita, přičemž uvažujeme velikost kontextu n. Chceme vytvořit slovní vektory o dimenzionalitě d. Dalším parametrem neuronové sítě je velikost druhé skryté vrstvy h.

Vstupem sítě jsou slova, která se nachází v kontextu; výstupem má být slovo, jehož kontext je popisován. Každé slovo na vstupu je zakódované pomocí *one-hot encoding*, tedy pomocí $|w|$ binárních proměnných, z nichž jedna je nastavená na 1, ostatní na 0 – dohromady tedy máme $n \cdot |w|$ binárních proměnných na vstupu. Na výstupu se nachází $|w|$ hodnot, každá z nich vyjadřuje pravděpodobnost, že se popisuje kontext určitého slova. Tyto pravděpodobnosti jsou přepočtené pomocí funkce *softmax*, jejich součet je proto vždy roven jedné.

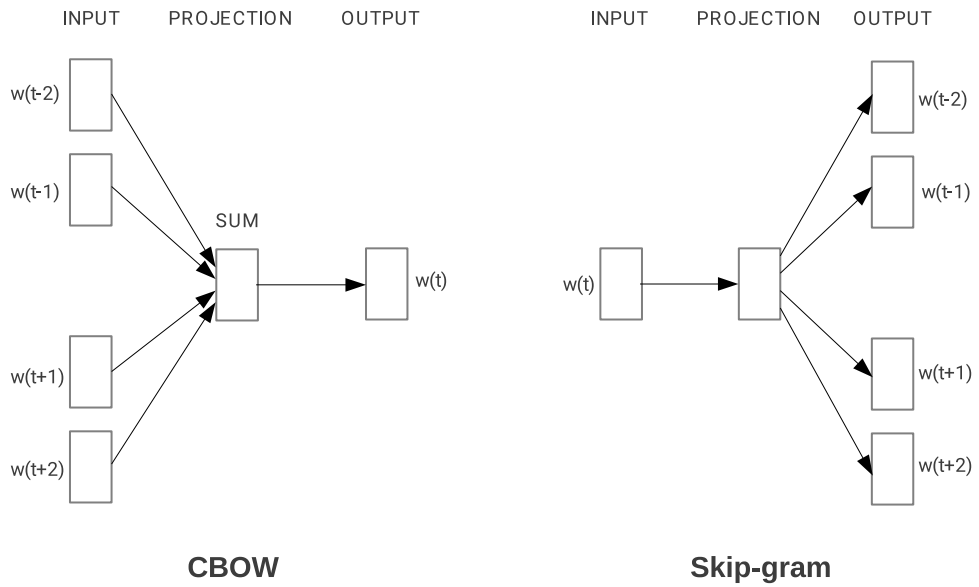
Síť se dále skládá ze dvou vnitřních vrstev. Vstupní proměnné sítě se používají na vstupu matice desetinných čísel C o velikosti $|w| \cdot d$. Obsahem této matice jsou slovní vektory. Ačkoliv na obrázku je C znázorněna několikrát vedle sebe, pořád jde jen o jednu matici s totožným obsahem. Výstupem vrstvy je $n-1$ slovních vektorů; tyto vektory jsou následně spojeny dohromady do vektoru o délce $(n-1) \cdot d$. Tento vektor se stává vstupem druhé skryté vrstvy, na jejímž vstupu je vynásoben maticí H o velikosti $h \cdot ((n-1) \cdot d)$; na výsledek je aplikován hyperbolický tangens. Přechodem z druhé skryté vrstvy do výsledku je matie U o velikosti $|w| \cdot h$.

Volné parametry, které je třeba natrénovat, obsahují matice C, H a U. Síť je trénována pomocí iterativní metody *stochastic gradient descent* [68].

CBOW a Skip-Gram

Mikolov a kol. [39] navrhli dva vylepšené modely založené na NNLM, jejichž hlavním cílem bylo dále zvýšit efektivitu trénování.

Model CBOW (*Continuous Bag-of-Words*) funguje podobně jako NNLM, odstraňuje však druhou skrytou vrstvu. Slovní vektory jsou namísto řazení vedle sebe sečteny a zprůměrovány – tato jednoduchá skrytá vrstva je poté přímo spojena s výstupní vrstvou. Pojmenování *bag-of-words* v názvu vyjadřuje, že jakmile se slovo vyskytne v kontextu, pak už nezáleží na jeho přesném pořadí. Druhý model, Skip-Gram, rovněž ruší nelineární skrytou vrstvu a navíc prohazuje funkci vstupní a výstupní vrstvy: vstupem je (jediné) slovo a výstupem jsou slova v kontextu. Schéma obou neuronových sítí jsou znázorněna na obrázku 4.4.

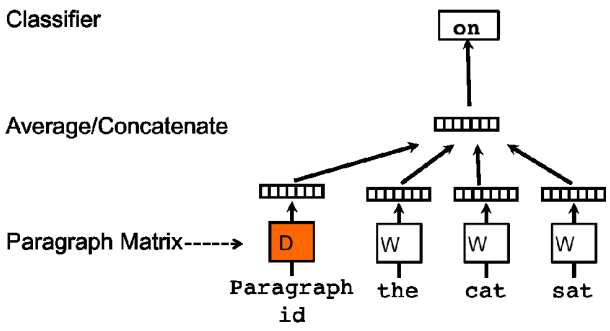


Obrázek 4.4: Vylepšení modelu NNLM podle Mikolova a kol. [39]: CBOW a Skip-Gram

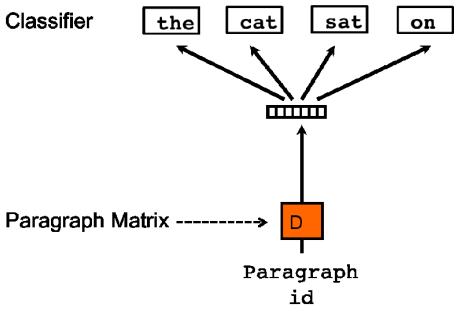
Navazující práce přinesly další vylepšení. Například Mikolov a kol. [40] vylepšují Skip-Gram pomocí hierarchické reprezentace slov (úspornější náhrada *one-hot encoding*) a vyřazováním často se vyskytujících slov.

FastText

Joulin a kol. [26] využili architekturu modelu CBOW jako základ pro klasifikátor textu *FastText*. Pro trénování klasifikace je potřeba sada dokumentů a množina kategorií; každý text je přiřazen do určité podmnožiny kategorií. Neuronová síť CBOW je upravena tak, že výstupem není pravděpodobnost určitého



Obrázek 4.5: Doc2Vec: struktura DM. [34]



Obrázek 4.6: Doc2Vec: struktura DBOW. [34]

slova, ale pravděpodobnost kategorie. Jako vylepšení se využívá hierarchické reprezentace slov [40].

Další důležitou vlastností klasifikátoru je využívání *znakových n-gramů*, n-tic znaků ve slově, místo samostatných slov: například slovo v dokumentu `kočka` do neuronové sítě vstupuje postupně jako `*ko`, `koč`, `očk`, `čka` a `ka*`, v případě, že je n nastavené na 3. V praxi se nepoužívá jediná možná hodnota n, ale určitý rozsah, například od 3 do 6. Díky využití znakových n-gramů mají význam i slova v testovacích datech, která se nevyskytovala v trénovacích datech – za předpokladu, že se některé jejich n-gramy jsou podobají n-gramům z trénovacích dat. Myšlenka znakových n-gramů byla představena v práci Bojanowského a kol. [12]

S publikací článku byla zároveň uvolněna i implementace[1]. Jak výzkum metody, tak implementace je podporována oddělením výzkumu společnosti Facebook. Právě tuto implementaci využijeme v experimentech.

Paragraph vectors / Doc2Vec

Jiný přístup ke klasifikaci textu na základě slovních vektorů zvolili Le a Mikolov [34]. Jejich práce nabízí dva modely založené na CBOW a Skip-Gram. Model DM (*Distributed Memory*) vychází z CBOW, přidává však na vstupu *paragraph ID*, označení kategorie (kategorií) dokumentu. Paragraph ID podle autorů „slouží jako jako paměť, umožňující si zapamatovat to, co schází z aktuálního kontextu“. Model DBOW (*Distributed Bag of Words*) vychází ze Skip-Gramu, vstupem sítě je však místo slova *paragraph ID*. Struktura obou neuronových sítí je znázorněna na obrázcích 4.6 a 4.5.

Metodu paragraph vectors, zvanou též Doc2Vec, rovněž využijeme v experimentech. Použijeme efektivní implementaci v knihovně Gensim [56].

4.4.3 Random Forests

Jako baseline metodu, používanou před masivním nástupem neuronových sítí, jsme zvolili metodu Random Forest, která se vnitřně skládá z velkého množství rozhodovacích stromů. Protože však metoda nedokáže pracovat přímo s textem dokumentů, je třeba všechny dokumenty převést na číselné vektory o určité dimenzi.

[1] `https://github.com/facebookresearch/fastText`

Bag-of-words

Existuje několik způsobů, jak převést textový dokument na číselný vektor. Populárním řešením, jak reprezentovat dokument, se stal model *Bag-of-words*. Vektory tohoto modelu mají dimenzi rovnou počtu různých slov, nacházejících se v dokumentech; každé slovo tedy odpovídá jedné dimenzi. Reprezentací textového dokumentu je takový vektor, že hodnota dimenze odpovídá počtu výskytu odpovídajícího slova v dokumentu. Možnou variantou je hodnoty ve vektoru omezit na hodnoty 1 a 0 – 1 v případě, že se slovo v dokumentu nachází, 0 v opačném případě. Tuto variantu nazýváme *unární Bag-of-words*.

Zásadní vlastností Bag-of-words je fakt, že nijak nepracuje s pořadím slov a s kontextem jednotlivých slov. Tím se zásadně liší od slovních vektorů (viz sekce 4.4.2). Výhodou bag-of-words je jednoduchost celého konceptu a velká efektivita výpočtu. Mezi nevýhody patří vysoká dimenzionalita: množina různorodých textových dokumentů může snadno obsahovat desetitisíce i statisíce různých slov. Vektory takovéto dimenzionality jsou pro další využití velmi špatně použitelné (problém *Curse of Dimensionality* [33]), proto je v dalším kroku třeba dimenzionalitu snížit.

Snižování dimenzionality

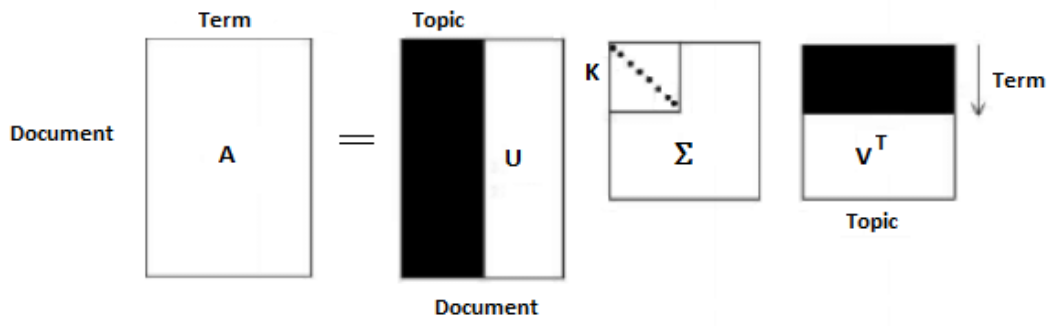
Jednou z možností snižování dimenzionality výsledku je vybrat pouze ty příznaky (dimenze vektoru), které jsou důležité pro rozpoznání textu. Metody se obvykle označují jako *feature selection methods* (viz např. Chandrashekar a Sahin [15]). V naší práci jsme se však rozhodli tyto postupy nevyužít. Důvodem je, že chceme snížit počet dimenzí na řádově stovky, kvůli časové a paměťové efektivitě a kvůli doporučení ze článků. Jednotlivé dimenze vektoru vzniklého pomocí Bag-of-words odpovídají slovům v dokumentu; pokud bychom použili na tento vektor snižování dimenzionality, zbyde nám informace jen o řádově stovkách slov, což považujeme za příliš hrubé.

Namísto toho využijeme metody *feature extraction* (*feature projection*). Obecně je cílem těchto metod transformovat vektory na vstupu do nižšího počtu dimenzí na výstupu. Transformace obvykle probíhá pomocí lineární projekce tak, že se vstupní vektor o dimenzi A zprava vynásobí *transformační maticí* o velikosti $A \times B$, kde B je dimenze výstupního vektoru.

Sada N dokumentů, převedená pomocí Bag-of-words do číselného vektoru o dimenzi D, je reprezentována jako matice $N \times D$; řádky matice odpovídají vektorům dokumentu. Tato matice je vstupem pro trénování metod PCA (*Principal component analysis*) [3] a LSI (*Latent semantic indexing*) [53].

LSI při trénování transformace provádí *singular value decomposition*, rozklad zdrojové matice M o velikosti $m \times n$ na součin matic U (velikost $m \times m$), Σ ($m \times n$) a V ($n \times n$). Matice Σ je diagonální. Reprezentaci vektorů o nižší dimenzi k získáme tak, že matice „ořízneme“: U na velikost $m \times k$, Σ na velikost $k \times k$ (ponecháme pouze k hodnot na diagonále) a V na velikost $k \times n$. Proces LSI je znázorněn na obrázku 4.7.

PCA rovněž provádí singular value decomposition, na rozdíl od LSI však pracuje s *kovarianční maticí*, která obsahuje hodnoty kovariance pro všechny dvojice dokumentů v trénovací sadě.



Obrázek 4.7: LSI: singular value decomposition a reprezentace dokumentů pomocí vektorů k o nižší dimenzi [24]

Ke snížení dimenzionality lze také použít *pravděpodobnostní metody*. Předpokladem těchto metod je, že sada dokumentů může být popsána pomocí určitého pravděpodobnostního modelu. Trénování metody pak znamená, že se hledají nejlepší možné parametry tohoto modelu. Příkladem může být Gaussian mixture model nebo Latent Dirichlet allocation (druhý z nich je přímo uzpůsoben pro použití v klasifikaci textu).

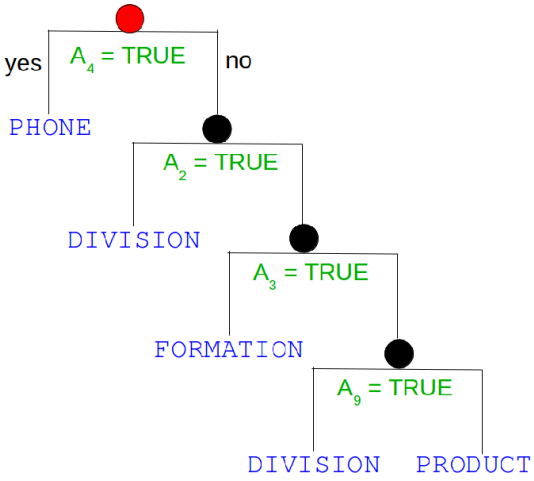
Problémem hledání pravděpodobnostních modelů je vysoká časová náročnost, proto jsme se je rozhodli nepoužít. Problémem PCA je paměťová náročnost: pro použití PCA je třeba v paměti vytvořit kovarianční matici, což znamená vysokou paměťovou složitost (kvadratickou vzhledem k počtu dokumentů), a proto jsme jej rovněž nevyužili. Pro snížení dimenzionality tedy bylo vybráno LSI.

Rozhodovací stromy

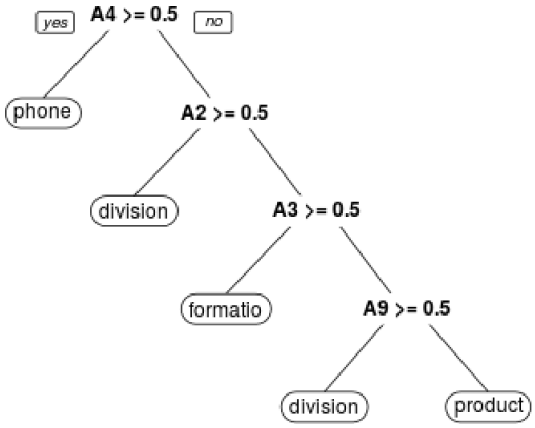
Rozhodovací strom je druh klasifikátoru, který je reprezentován binárním stromem. Vstupem do tohoto binárního stromu je vektor příznaků, v našem případě číselný vektor, jehož dimenzionalita byla snížena pomocí LSI. Výstupem je seznam kategorií, do nichž dokument reprezentovaný vektorem náleží. Každý vnitřní vrchol stromu obsahuje podmínku na hodnotu konkrétní dimenze vektoru. Každý list obsahuje výstupní seznam kategorií. Klasifikace dokumentu probíhá tak, že začneme v kořeni binárního stromu a postupně se přesouváme do dalších vrcholů níže v hierarchii stromu, dokud nenarazíme na list. V každém vnitřním vrcholu zkontrolujeme podmínku; pokud ji vektor dokumentu splňuje, přesuneme se do levého podstromu, pokud ne, přesuneme se do pravého podstromu. Příklad rozhodovacích stromů je na obrázcích 4.8 a 4.9.

Trénování (stavba) rozhodovacího stromu je rekurzivní problém. Během trénování je s každým vrcholem spojena určitá podmnožina trénovacích dokumentů. Strom tvoříme od kořene, se kterým spojíme všechny trénovací dokumenty. Každý vrchol poté zpracujeme takto: buď ho prohlásíme za list a přiřadíme mu jako výsledek podmnožinu kategorií, která je nejčastější v dokumentech spojených s tímto vrcholem; nebo vytvoříme nové dva vrcholy, které budou podřazené současnému vrcholu, rozdělíme dokumenty na dvě vhodné části, přiřadíme je vrcholům a postupujeme rekurzivně. Obrázek 4.10 znázorňuje proces stavby stromu.

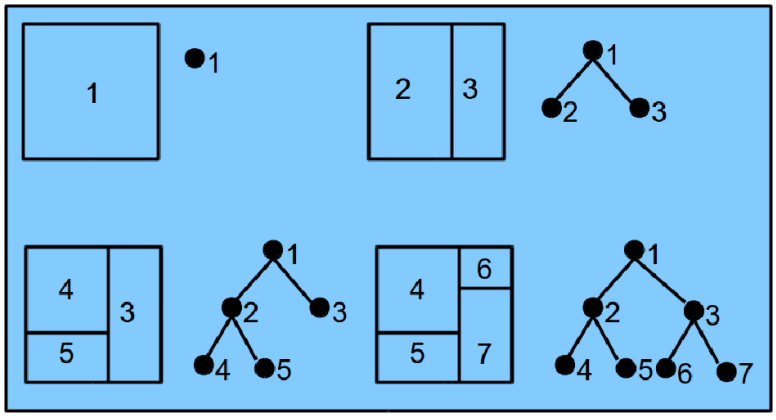
V tomto algoritmu musíme vyřešit, kdy zastavit „rozrůstání stromu“ (tvoření nových vrcholů), jak dělit dokumenty a jakou podmínku zvolit. Rozrůstání stromu



Obrázek 4.8: Příklad rozhodovacího stromu: prvky vektoru jsou binární proměnné. [22]



Obrázek 4.9: Příklad rozhodovacího stromu: prvky vektoru jsou číselné proměnné. [22]



Obrázek 4.10: Příklad stavby rozhodovacího stromu. Čtverec představuje datovou sadu a její dělení. [22]

zastavíme v případě, že dokumenty spojené s aktuálním vrcholem mají všechny stejnou množinu kategorií, případně na základě jiných podmínek, například celkového množství dokumentů spojených s vrcholem. Dokumenty se snažíme rozdělit na dvě sady pomocí takové podmínky, aby kategorie obou sad byly co nejodlišnější (došlo ke snížení míry entropie).

Random forests

Random forests je metoda klasifikace, která se snaží vyřešit slabá místa rozhodovacích stomů. Jedním ze slabých míst je nestabilita algoritmu pro trénování: použití jiné podmínky pro rozdělení dokumentů vede k vytvoření úplně jiného stromu. Dalším slabým místem, v pozdějších fázích trénování, je možnost použít pouze malou část datové sady. Metoda *random forests* funguje tak, že namísto jednoho rozhodovacího stromu jich vytvoří větší množství, každý strom na náhodně vybrané podmnožině dat. Klasifikace pak proběhne na všech stromech najednou a je vrácen nejčastější výsledek.

Implementace

Pro Bag-of-words a snížení dimenzionality jsme použili části knihovny Gensim [56], která funguje efektivně i při práci s většími datovými sadami. Jako implementaci random forests jsme vybrali knihovnu *scikit-learn*.

4.4.4 Klíčová slova

Jak už bylo zmíněno v diskuzi analýzy (sekce 3.3), máme zájem vyzkoušet rozpoznávání kategorií pouze na základě klíčových slov. Tento model má výhodu ve snadné interpretovatelnosti a flexibilitě; myšlenka popisu každé kategorie pomocí klíčových slov je velmi snadno pochopitelná. Je ale třeba najít správný postup trénování, to znamená extrahování klíčových slov z otagované datové sady. Aby klíčová slova zůstala analyzovatelná, musíme také omezit jejich množství.

Statistika významu slov a TF-IDF

Vyfiltrování slov významných pro určení kategorie jsme částečně provedli již během během předzpracování (viz sekce 4.3). Pro získání kvalitní a kompaktní sady klíčových slov to ale není dostatečné. Jak bychom mohli blíže určit významnost slova? Pro zjednodušení na chvíli předpokládejme, že máme k dispozici sadu textových dokumentů bez jakéhokoliv otagování. Nemáme velký zájem o slova vyskytující se ve většině dokumentů, budeme je považovat za nevýznamná. Naopak čím unikátnější slovo je, tím větší je pravděpodobnost, že jde o významné slovo. V rámci jediného dokumentu si můžeme snadno spočítat, kolikrát se v něm nějaké slovo vyskytuje – čím větší výskyt, tím významnější slovo bude.

Toto intuitivní pozorování, které jsme si právě uvedli, je základní myšlenkou metriky TF-IDF (*term frequency–inverse document frequency*). Mějme množinu dokumentů D a množinu slov T; TF-IDF přiřadí každé dvojici slova $t \in T$ a dokumentu $d \in D$ hodnotu

$$\mathrm{idf}_{t,d} = \mathrm{tf}_{t,d} \times \mathrm{idf}_t \tag{4.3}$$

přičemž významy jednotlivých činitelů jsou následující:

- $\mathrm{tf}_{t,d}$ je *term frequency*: vyjadřuje význam slova v rámci jednoho dokumentu. Hodnota typicky roste se zvyšujícím se výskytem slova t v dokumentu d.
- idf_t je *inverse document frequency*: vyjadřuje význam slova v rámci celé sady dokumentů. Hodnota typicky klesá se zvyšujícím se výskytem slova t napříč všemi dokumenty.

Manning a kol. [37] popisují různé způsoby výpočtů obou činitelů, například TF bývá často logaritmizováno, aby se zabránil jeho přílišný nárůst.

Úpravy TF-IDF pro kategorizaci textu

TF-IDF v základu není určené pro klasifikaci textu, pro naši situaci proto budeme muset provést úpravy postupu výpočtu.

Zadefinujme si *naddokument kategorie*, sloučený text všech dokumentů, kterým je tato kategorie přiřazena. Teprve tyto naddokumenty budou tvořit vstup TF-IDF. Pokud má dokument přiřazených více kategorií, pak bude vložen do více naddokumentů zároveň. Uvědomme si, že pro extrakci klíčových slov pro jednotlivé kategorie je tento postup výhodný: čím více se určité slovo vyskytuje v dokumentech napříč různými kategoriemi, tím méně je pravděpodobné, že se jedná o klíčové slovo; čím více se objevuje v rámci naddokumentu jediné kategorie, tím více vzrůstá pravděpodobnost, že se jedná o klíčové slovo.

Počítat TF jako prostý počet výskytů slova v naddokumentu je však problematické. Vzhledem k velmi nevyrovnanému počtu dokumentů pro kategorie (viz sekce 3.1.1) budou mít naddokumenty velmi nevyrovnanou délku. Také bychom chtěli, aby se TF zvyšovalo v případě, že slovo bude uvedené v *různých* dokumentech – pokud ale budeme počítat výskyty v naddokumentu, může se stát, že v některém delším dokumentu bude slovo uvedené víckrát. Proto při výpočtu TF budeme uvažovat dvě vstupní hodnoty: celkový počet dokumentů v naddokumentu a počet dokumentů v naddokumentu, kde se vyskytuje příslušné slovo.

IDF ponecháme bez úprav; rozhodně však chceme přiřadit velmi nízkou hodnotu slovům, které se vyskytují ve většině kategorií. V takovém případě totiž pravděpodobně jde o *stop-slova*, běžně se vyskytující v jakémkoliv textu.

Po dokončení výpočtu vybereme pro každou kategorii slova s nejvyššími hodnotami TF-IDF, která prohlásíme za klíčová slova kategorie. Konkrétní hodnoty TF-IDF nebudeme dále používat, i když by se to nabízelo; předběžné testování ukázalo, že je mnohem efektivnější uvažovat extrahovaná slova jako rovnocenná.

Zkoumané varianty TF-IDF

V experimentech zkusíme počítat TF několika různými způsoby. V následujících vzorcích označuje d celkový počet dokumentů v naddokumentu a n počet dokumentů s výskytem slova:

- `log`: $\mathrm{tf} = \log_2(1+n)$
- `sqrt_ratio`: $\mathrm{tf} = \sqrt{\frac{n}{d}}$

- `over0.005`: $\mathrm{tf} = \begin{cases} \frac{n}{d} & \frac{n}{d} > 0{,}005 \\ 0 & \frac{n}{d} \leq 0{,}005 \end{cases}$

Ve variantě `log` vůbec neuvažujeme celkový počet dokumentů v kategorii. Základem metody `sqrt_ratio` je jednoduchý poměr počtu dokumentů s výskytem slova a celkového počtu dokumentů; kvůli odmocnině však hodnota roste rychleji při zvyšování n z nízkých hodnot. Cílem varianty `over0.005` je úplně vyřadit slova, která se vyskytují v příliš malém množství dokumentů, ale mohla by se kvůli jinak vysokému IDF dostat do výběru.

Rovněž výpočet IDF bude mít několik variant. Zde t označuje celkový počet kategorií a n počet kategorií, kde se slovo alespoň jednou vyskytuje.

Zápis $\mathrm{lin}[x_1,y_1,x_2,y_2]$ označuje lineární funkci, která má v bodě x_1 hodnotu y_1 a v x_2 hodnotu y_2. Zápis $I[\text{podminka}]$ je indikátor, při splnění podmínky je hodnota 1, jinak 0.

- `none`: $\mathrm{idf} = 1$
- `lin910`: $\mathrm{idf} = \begin{cases} \mathrm{lin}[1{,}1{,}\frac{9}{10} \cdot t + 1{,}0] & \frac{n}{t} < \frac{9}{10} \\ 0 & \frac{n}{t} \geq \frac{9}{10} \end{cases}$
- `sharp910`: $\mathrm{idf} = I[\frac{n}{t} < \frac{9}{10}]$

Varianta `none` znamená vypnutí IDF. Použitím `none` bychom chtěli změřit změnu výkonnosti při použití IDF. Možnost `lin910` vyřazuje slovo, pokud se nachází ve více než devíti desetinách kategorií; přiřazují slovu hodnotu 1, pokud se nachází pouze v jediné kategorii; při výskytu v dalších kategoriích se hodnota lineárně snižuje. Poslední varianta `sharp910` buď slovo vyřadí, nebo mu přiřadí hodnotu 1.

Vyzkoušíme rovněž možnost, kdy po dokončení výpočtu ponecháme z každé kategorie pouze 1 000 slov s nejvyšším ohodnocením TF-IDF. Zároveň se v této sadě slov nebude při predikci používat hodnota TF-IDF, ale všechna slova budou mít hodnotu 1.

Implementace

Vzhledem k jednoduchosti počítání všech variant TF-IDF a našim úpravám pro klasifikaci textu jsme použili vlastní implementaci v Pythonu.

4.5 Transfer learning

Vzhledem k tomu, že trénovací a testovací sada dokumentů se v našem zadání liší, používáme *transfer learning*, konkrétně *transduktivní učení*. Toto odpovídá definici podle Pana a Yanga [52]. Zakázky jsme pro trénování modelů použili z toho důvodu, že jsou již zařazené do kategorií (převedením CPV kódů), jejich množství je srovnatelné s množstvím smluv a jejich texty souvisí s texty smluv. Alternativou k tomuto postupu by například bylo vzít podmnožinu smluv, ručně označit smlouvy v této podmnožině správnými kategoriemi, a tento malý vzorek použít jako trénovací sadu. Nevýhodou je pochopitelně pracnost a náchylnost

k chybám, ovšem také nutnost označit opravdu velké množství smluv. Vzhledem k vysokému počtu kategorií jich musí být minimálně 104, ale spíše vhodný násobek tohoto čísla. Z těchto důvodů jsme se rozhodli vydat cestou transfer learningu.

Musíme ale uvážit, zda vybrané metody klasifikace není třeba nějak upravit. Nedokázali jsme však k tomuto tématu dohledat dostatek použitelných a ověřených informací. Nenalezli jsme práce, které by zkoumaly vliv transfer learningu na výkonnost konkrétních klasifikátorů, které používáme. Práce Pana a Yanga v případě transduktivního učení hovoří o vlivu pravděpodobnostního rozložení kategorií v datové sadě a situaci, kdy trénovací a testovací sada mají toto rozlišení odlišné; z důvodu neznalosti skutečného rozložení kategorií v datové sadě smluv však nemůžeme uvedený postup využít.

Skutečné chování námi vybraných klasifikátorů při různých sadách trénovacích a testovacích dat je tedy zatím neznámé. Klasifikátory nicméně chceme otestovat jak na veřejných zakázkách (respektive testovací sadě veřejných zakázek, kterou jsme nepoužili pro trénování), tak na smlouvách (malé podmnožině smluv, kterou ručně označíme správnými kategoriemi). Proto pokud některý z klasifikátorů funguje výborně pro trénovací a testovací data ze stejné domény, ale selhává pro data z různých domén, zjistíme to při analýze výsledků experimentů.

5. Experimenty

Tématem této kapitoly je otestování našeho řešení, porovnání výkonnosti jednotlivých metod klasifikace a diskuze výsledků. Nejprve si představíme naše testovací prostředí, shrneme všechny zkoumané varianty testování a rozhodneme, na základě kterých metrik posuzovat výkonnost klasifikace. V dalších částech seznámíme čtenáře s výsledky a provedeme řazení variant testování podle výkonnosti. Ve shrnutí pak výsledky analyzujeme a vybereme nejvhodnější řešení pro praktickou klasifikaci na webu Hlídače státu.

5.1 Modifikace datových sad

Jedním z našich cílů je otestovat výkonnost klasifikátorů jak na smlouvách, tak na veřejných zakázkách. Z tohoto důvodu si datovou sadu veřejných zakázek podrozdělíme na trénovací a testovací sadu: k trénování klasifikátorů využijeme 80 procent zakázek, zbylých 20 procent bude sloužit k testování. Dále jsme z datové sady smluv vybrali náhodně 204 smluv a ručně jsme jim přiřadili správné kategorie. Máme tedy k dispozici tyto datové sady:

- *trénovací sada veřejných zakázek* – 80 procent zakázek – použije se k trénování klasifikátorů,
- *testovací sada veřejných zakázek* – 20 procent zakázek – použije se k testování,
- *testovací sada smluv* – 204 ručně označených smluv – také se použije k testování,
- zbylé, neoznačené smlouvy – nebudeme používat.

5.2 Testovací scénáře

Každý testovací scénář je určený kombinací tří faktorů: metody klasifikace, parametrů metody klasifikace a datové sady. Tyto jednotlivé faktory nyní podrobněji popíšeme.

V rámci testů provedeme takzvaný *grid search*: vyzkoušíme každou možnou kombinaci faktorů a ohodnotíme její výkonnost. Tuto možnost jsme vybrali z toho důvodu, že mezi faktory není žádná zjevná závislost – jakákoliv kombinace nám může teoreticky vrátit nejlepší možný výsledek. Nevýhodou tohoto přístupu samozřejmě je to, že s jakoukoliv další přidanou možností rychle narůstá celkový počet scénářů.

Metody klasifikace a jejich parametry

V tabulce 5.1 jsou uvedeny metody klasifikace spolu s parametry, které testujeme.

FastText	dimenze vektorů	100, 500
	počet epoch trénování	3, 5, 10
	délka znakových n-gramů	1, 3, 10
Doc2Vec	dimenze vektorů kategorií	100, 500
	model	DM, DBOW
	velikost kontextu slova	1, 3, 10
	počet epoch trénování	3, 5
Náhodné stromy	reprezentace dokumentu	BoW, unární BoW, TF-IDF
	dimenze LSI	100, 500
	počet náhodných stromů	100, 500
	maximální počet dokumentů v uzlu	10, 100
Klíčová slova	metoda IDF	log, sqrt_ratio, over0.005
	metoda TF	none, lin910, sharp910
	postprocessing slov	postnone, post1000

Tabulka 5.1: Metody klasifikace a parametry, které testujeme

Datové sady k otestování

Jak jsme již zmínili v sekci 5.1, máme dvě testovací sady: veřejné zakázky (20 procent z celkového počtu) a 204 označených smluv.

5.3 Testovací prostředí

K testování jsme využili školní výpočetní cluster *gpulab*, konkrétně jeho stroj *dw04*. Obsahuje procesor Intel Xeon E5-2660 v2, frekvence 2,20 GHz, 40 jader. K dispozici má 248 GB operační paměti. Na clusteru funguje plánovač úloh Slurm.

Veškerý náš zdrojový kód pro testování byl implementován v jazyce Python, konkrétně verzi 3.7. Konkrétnější informace o struktuře zdrojových kódů lze nalézt v příloze C.

5.4 Výběr metrik

Výkon klasifikátorů je možné hodnotit pomocí různých hodnotících funkcí – metrik. Abychom mohli vybrat ty metriky, které jsou nejužitečnější pro náš problém, musíme si uvědomit, jak budeme klasifikaci smluv ve výsledku používat. Vybrané metriky by s těmito případy použití měly korelovat: čím lépe hodnotí klasifikátor, tím kvalitnější by mělo být jeho použití v praxi.

Jak již bylo zmíněno v zadání (kapitola 2), natrénovaný klasifikátor smluv se na webu Hlídače státu bude používat ve dvou scénářích. Pro detail zobrazené smlouvy zobrazí predikované kategorie, přičemž každá kategorie může být uvedena s pravděpodobností, že do ní smlouva náleží. Na webu bude možné použít vyhledávání podle kategorií a seznam smluv, který bude vrácen uživateli, by měl být co nejlépe seřazený podle relevance; to znamená, že na nejvyšších příčkách by měly být smlouvy vybrané kategorie, ale s prohlížením dalších a dalších výsledků je možné zobrazit i jiné smlouvy. Toto chování odpovídá fulltextovým vyhledávačům na webu: je obvyklé, že například třetí stránka s výsledky a další následující stránky zobrazí téměř nerelevantní odkazy.

5.4.1 Kategorie metrik

Kvalitní souhrn a porovnání různých metrik používaných ve strojovém učení nabízí práce Ferriho a kol. [17] Autoři je rozdělili do tří různých kategorií, které zde uvádíme.

- metriky měřící chybu *kvalitativně*: předpokládá se, že klasifikátor predikuje pro každou kategorii pouze hodnoty 1 (náleží) nebo 0 (nenáleží); stejně tak skutečné hodnoty mohou nabývat pouze těchto dvou hodnot. Predikce klasifikátoru je proto vždy jednoznačně správná, nebo špatná; použitá metrika pracuje s četností správných a špatných predikcí.

- metriky měřící *statistické rozložení chyby*: hodnotí se odchylka predikované hodnoty a skutečné hodnoty.

- metriky posuzující *seřazení predikovaných hodnot*, například: pokud skutečná hodnota pro dokument A je větší, než skutečná hodnota dokumentu B, pak i predikovaná hodnota pro A by měla být větší, než predikovaná hodnota B.

V našem případě jsou skutečné hodnoty (zda dokument náleží do kategorie) vždy buď 0, nebo 1; predikované hodnoty mohou být libovolné nezáporné – například u FastTextu vždy v intervalu [0,1], ale při rozpoznávání klíčových slov jde o nezáporná celá čísla. Klasifikátory, jako například FastText, mají ještě jednu problematickou vlastnost: závěrečná úprava hodnot predikce pomocí funkce *softmax* zařídí, že součet predikovaných hodnot je vždy 1. Pokud klasifikátor založený na FastTextu například usoudí, že dokument náleží pouze do jedné ze všech kategorií, může být hodnota pro tuto kategorii 0,99 (0,01 je marginální hodnota pro zbylé kategorie); pokud ale dokument rovnocenně zařadí do tří kategorií, ohodnocení pro každou tuto kategorii se sníží na 0,3.

Z uvedených důvodů vyplývá, že porovnávat hodnoty absolutně je velmi problematické, a to i v rámci jedné klasifikační metody. Můžeme se řídit pouze jediným pravidlem: pro výsledky klasifikace konkrétní smlouvy platí, že čím vyšší ohodnocení je přiřazeno kategorii, tím větší je podle klasifikátoru pravděpodobnost, že smlouva do dané kategorie náleží.

Rozhodli jsme se proto vůbec nepoužít metriky měřící statistické rozložení chyby, protože pracují s odchylkou dvou absolutních hodnot. Jde například o funkci *Mean Average Error* (*MAE*), průměrnou absolutní hodnotu rozdílu skutečné hodnoty a predikované hodnoty. Jak seřazení výsledků při vyhledávání, tak predikci kategorií u konkrétní smlouvy změříme pomocí metrik z ostatních dvou kategorií.

5.4.2 Použité značení

V dalších odstavcích zadefinujeme několik různých metrik a porovnáme jejich význam. Zavedeme si následující značení:

- D je množina testovaných dokumentů,

- p představuje klasifikátor,

- C je množina kategorií, kterou používá klasifikátor p,
- $p_c(d) \in [0,1]$ je predikce klasifikátoru p pro dokument $d \in D$ a kategorii $c \in C$,
- $r_c(d) \in \mathbb{R}_0^+$ označuje skutečnou hodnotu pro dokument $d \in D$ a kategorii $c \in C$.
- $D_0^c = \{d \in D, r_c(d) = 0\}$
- $D_1^c = \{d \in D, r_c(d) = 1\}$
- $rank(x_0, \{(x,i), x \in X, i \in \mathbb{R}\}), x_0 \in X$ znamená, že seřadíme dvojice v množině podle hodnoty i sestupně a vrátíme pořadí té dvojice, kde $x_0 = x$. Pořadí číslujeme od 1.

5.4.3 Metrika pro seřazení výsledků

Pro seřazení výsledků chceme jistě použít metodu měření, posuzující řazení daných hodnot. Budeme posuzovat schopnost klasifikátoru řadit zvlášť pro každou kategorii – je totiž možné, že pro některé kategorie bude řazení velmi kvalitní, pro jiné horší. Následující definice přebíráme z práce Wu a Zhou [66], avšak na rozdíl od autorů je omezujeme vždy pouze na jedinou kategorii:

- *AUC (area under curve)*: popisuje četnost dvojic dokumentů, jejichž predikované hodnoty pro kategorii c jsou správně seřazené:

$$AUC(p,c) = \frac{\sum_{d_0 \in D_0^c} \sum_{d_1 \in D_1^c} I[p_c(d_0) < p_c(d_1)]}{|D_0^c||D_1^c|} \tag{5.1}$$

- *Ranking loss*: definované velmi podobně jako AUC, ale popisuje četnost dvojic dokumentů, jejichž predikované hodnoty jsou špatně seřazené:

$$Rloss(p,c) = \frac{\sum_{d_0 \in D_0^c} \sum_{d_1 \in D_1^c} I[p_c(d_0) > p_c(d_1)]}{|D_0^c||D_1^c|} \tag{5.2}$$

- *Coverage*: odpovídá pořadí posledního dokumentu, který náleží do kategorie:

$$Coverage(p,c) = \max_{d \in D_1^c} rank(d, \{(d_0, p_c(d_0)), d_0 \in D\}) \tag{5.3}$$

Využití Coverage je problematické z toho důvodu, že uvažuje pouze pořadí posledního relevantního dokumentu, což je pro posouzení výkonnosti klasifikátoru příliš hrubé. Snadno se může stát, že ačkoliv klasifikátor umísťuje naprostou většinu relevantních dokumentů na začátek seznamu (tj. dává jim vysoké ohodnocení), kvůli jediné nerozpoznané relevantní smlouvě se hodnocení podle Coverage velmi zhorší. Ranking loss a AUC měří výkonnost stejným způsobem, pouze obráceně (vyšší hodnota je pro AUC lepší, pro Ranking loss horší). Vybrali jsme však metriku AUC, neboť se v literatuře používá častěji.

5.4.4 Metrika pro ohodnocení smlouvy

Pro posouzení výkonnosti zařazení jednotlivé smlouvy do správných kategorií musíme použít jiný přístup než v předchozím problému. Pro hodnocení seřazení výsledků jsme provedli klasifikaci pro všechny smlouvy v testovací sadě a vybrali výsledek pro jedinou kategorii; zde naopak pracujeme s klasifikacemi smlouvy pro všechny kategorie najednou.

Pro výsledky klasifikace jedné smlouvy platí, že čím vyšší hodnocení má kategorie, tím vyšší je podle klasifikátoru pravděpodobnost, že smlouva do kategorie náleží. Mohli bychom tedy využít některou z metrik představených v předchozí části: pouze bychom použili řadu predikovaných hodnot ne pro jednu kategorii, ale pro jednu smlouvu.

Problém tohoto přístupu je však takový, že neodpovídá skutečnému použití na webu Hlídače státu. Metriky jako AUC a podobné totiž pracují s předpokladem, že v detailu smlouvy zobrazíme nenulové hodnoty predikce pro všechny kategorie – to by však potenciálně znamenalo až příliš velké množství dat. V praxi se chceme omezit na zobrazení malého počtu kategorií s nejvyšším ohodnocením. Protože nedokážeme snadno posoudit, které kladné hodnoty predikce považovat za *relevantní* a které za *nerelevantní*, použijeme fixní počet zobrazených kategorií. Konkrétně pět kategorií s nejvyšším ohodnocením – toto číslo jsme vybrali z toho důvodu, že zhruba odpovídá nadprůměrnému počtu kategorií ve veřejných zakázkách a je stále přehledné pro uživatele.

Tento postup nám navíc dovolí použít i metriky měřící chybu kvalitativně: predikované hodnoty pro kategorie můžeme převést na 1, pokud se vyskytují mezi pěti nejlepšími, jinak je převedeme na 0.

Rozšíříme notaci o následující definice:

- $p'_c(d)$ predikuje 1 nebo 0, podle toho, zda je c mezi pěti nejlepšími

$$p'_c(d) = I[rank(c, \{(c_0, p_{c_0}(d)), c_0 \in C\}) \leq 5] \tag{5.4}$$

- $TP(p, d) = \{c, c \in C, r_c(d) = 1, p'_c(d) = 1\}$
- $TN(p, d) = \{c, c \in C, r_c(d) = 0, p'_c(d) = 0\}$
- $FP(p, d) = \{c, c \in C, r_c(d) = 0, p'_c(d) = 1\}$
- $FN(p, d) = \{c, c \in C, r_c(d) = 1, p'_c(d) = 0\}$

Následující definice metrik měřících chybu kvalitativně jsme čerpali z Ferriho a kol. [17]:

- *Accuracy*: jde o poměr počtu správně predikovaných kategorií ke všem kategoriím:

$$Accuracy(p, d) = \frac{|TP| + |TN|}{|C|} \tag{5.5}$$

- *Precision*: vyjadřuje, jaké množství z predikovaných kategorií je relevantní:

$$Precision(p, d) = \frac{|TP|}{|TP| + |FP|} \tag{5.6}$$

- *Recall*: vyjadřuje, jaké množství relevantních kategorií bylo predikováno:

$$Recall(p, d) = \frac{|TP|}{|TP| + |FN|} \tag{5.7}$$

- *F-measure*: jde o geometrický průměr precision a recall:

$$F - measure(p, d) = \frac{2 \cdot Precision(p, d) \cdot Recall(p, d)}{Precision(p, d) + Recall(p, d)} \tag{5.8}$$

Ačkoliv je accuracy snadno pochopitelná, více informací o výkonnosti klasifikátoru dokážeme zjistit z hodnot precision a recall. Tyto dvě metriky dokáží posoudit různé kvality klasifikátoru: za prvé schopnost predikovat relevantní kategorie, za druhé vyřadit nerelevantní kategorii. Použijeme proto tyto dvě metriky. Jako doplnění budeme z těchto dvou metrik počítat F-measure, která jejich výsledek vhodným způsobem kombinuje. Použití geometrického průměru znamená, že hodnota metriky se bude spíše blížit k horšímu výsledku zdrojových metrik – pokud například bude precision vysoká a recall nízká, bude F-measure rovněž nízké.

Takto popsané metriky však popisují kvalitu klasifikace jen pro jednu smlouvu. Výsledky pro všechny smlouvy v testovací sadě agregujeme zprůměrováním, podobně jako uvádí Ferri a kol. [17] i Wu a Zhou [66].

5.5 Popis výsledků

V této sekci se budeme věnovat výsledkům testování klasifikátorů. Nejprve posoudíme vliv parametrů modelů na úspěšnost klasifikace. Dále vybereme nejlepší klasifikátory podle metriky F-measure a také podle AUC.

5.5.1 Vliv parametrů

V první řadě jsme zkoumali vliv parametrů modelů na výsledek klasifikace. Pro každý parametr a jeho možnou hodnotu jsme vyfiltrovali výsledky testování, kde se tento parametr s příslušnou hodnotou vyskytoval. Z těchto výsledků jsme spočetli průměrnou F-measure. Rozdělili jsme také výsledky podle datových sad. Grafy po jednotlivých metodách klasifikace a parametrech se nachází na obrázku 5.1 pro veřejné zakázky a obrázku 5.2 pro smlouvy.

Pro metodu random forests je pro obě datové sady nejdůležitější parametr `base_model`, označující reprezentaci slov. V obou případech je nejvýhodnější použití TF-IDF pro reprezentaci; pro veřejné zakázky je unární Bag-of-Words (každé nalezené slovo se započítává pouze jednou) výhodnější než standardní Bag-of-Words, avšak pro smlouvy tyto dvě hodnoty parametru nepřinášejí viditelný rozdíl. Zvyšování parametru `min_samples_split` přináší horší výsledky pro veřejné zakázky, ale pro smlouvy je pokles nepatrný. Zbývající parametry, `dim` (dimenze LSI) a `num_trees` (počet použitých náhodných stromů) nepřinášejí významné rozdíly ve výkonnosti.

Pro metodu Doc2Vec je zásadní parametr `algo`, označující použitou strukturu neuronové sítě; použití DBOW přináší výrazně lepší výsledky než DM. Důležitost tohoto parametru není překvapivá, vzhledem k tomu, že přepíná mezi dvěma

různými neuronovými sítěmi. Rovněž důležitý je parametr `epoch`, označující počet epoch při trénování klasifikátoru. S přibývajícím počtem epoch je obvykle spojena větší úspěšnost modelu při testování, avšak hrozí riziko *přetrénování (overtraining)*, kdy model funguje správně pro trénovací data, ale špatně pro testovací data. F-measure pro hodnoty epoch menší nebo rovné 10 stoupá, avšak mezi hodnotami 10 a 20 buď stagnuje (veřejné zakázky) nebo dokonce klesá (smlouvy). Toto chování může napovídat, že klasifikátor Doc2Vec s parametrem `epoch` rovným 20 je přetrénovaný. Pro parametr `window_size` je v obou případech nejlepší hodnota 3. Zvyšování dimenze vektorů snižuje F-measure, obzvlášť pro veřejné zakázky.

Výkonnost metody FastText velmi závisí na parametru `epoch`, jehož význam je totožný jako pro metodu Doc2Vec. Obzvlášť pro hodnoty do 10 přináší použití vyššího množství iterací při trénování zvýšení výkonnosti. Nad hodnotu 10 se toto zvyšování zpomaluje, avšak je stále patrné. Podobně jako pro Doc2Vec, i zde zvyšování dimenze vektorů zhoršuje F-measure.

Nejodlišnější výsledky podle použité datové sady přinesla metoda klíčových slov. Pro parametr `postprocess` jsou naměřené F-measures vyloženě protichůdné: zatímco pro veřejné zakázky se jeví jako výhodnější použít hodnotu `postnone` (nijak neupravovat sady slov pro kategorie), pro smlouvy přinese zlepšení hodnota `post1000` (z každé kategorie ponechat pouze 1 000 nejdůležitějších slov). Tato vlastnost by mohla poukazovat na přetrénování modelu s hodnotou `postnone`, kdy takový model nedokáže dostatečně generalizovat při použití odlišné datové sady. Podobně rozporuplný je parametr `idf`, jehož naměřené výsledky nicméně souvisí s předešlým parametrem. Hodnota `sharp910`, kdy dojde k zahození slov, vyskytujících se ve více 9/10 všech trénovacích dokumentů, přináší špatné výsledky pro zakázky, avšak dobré výsledky pro smlouvy. V obou případech nicméně horší výsledky přináší použití hodnoty `none`, kdy se IDF vůbec nepoužije. Pro parametr `tf` je v případě zakázek nejvýhodnější hodnota `log`, avšak v případě smluv hodnota `sqrt_ratio`.

5.5.2 Srovnání klasifikátorů podle F-measure

Pro každou metodu jsme vybrali pět nejlepších klasifikátorů podle hodnoty F-measure; pro veřejné zakázky jsou tyto klasifikátory uvedené v tabulce 5.2, pro smlouvy v tabulce 5.3. V tabulkách jsou uvedené také výsledky precision a recall, v naprosté většině případů však nižší F-measure znamená také nižší hodnoty precision i recall. Nedošlo k případu, kdy by například vyšší hodnota precision byla „vykoupena“ nižší hodnotou recall. Obecně nízké hodnoty precision byly očekávané; vzhledem k pevnému počtu uvažovaných kategorií ve výsledku klasifikace dokumentu se snadno mohlo stát, že množství skutečných kategorií dokumentu bylo nižší než tento počet. Hlavním cílem metrik však bylo porovnat výkonnost klasifikátorů mezi sebou.

Tabulky potvrzují naše pozorování z minulé sekce: v případě random forests je hodnota `base_model` téměř vždy `tfidf`, pro Doc2Vec je `algo` rovno `dbow`, FastText konzistentně drží parametr `epoch` na hodnotě 20. Pouze metoda klíčových slov má své parametry nejednoznačné.

Jednoznačně nejhorší výsledek zaznamenaly random forests – F-measure pěti nejlepších modelů založených na random forests bylo horší, než F-measure nej-

met	parametry metody	F-meas	prec	recall
FT	**dim: 100, epoch: 20, ws: 3**	**0.599**	**0.453**	**0.883**
FT	dim: 100, epoch: 20, ws: 1	0.599	0.453	0.883
FT	dim: 100, epoch: 20, ws: 10	0.599	0.453	0.882
FT	dim: 500, epoch: 20, ws: 1	0.597	0.452	0.880
FT	dim: 500, epoch: 20, ws: 10	0.597	0.452	0.880
KW	**idf: lin910, tf: log, post: postnone**	**0.597**	**0.453**	**0.874**
D2V	**dim: 500, algo: dbow, ws: 10, epoch: 10**	**0.569**	**0.431**	**0.836**
D2V	dim: 500, algo: dbow, ws: 1, epoch: 10	0.569	0.431	0.836
D2V	dim: 500, algo: dbow, ws: 3, epoch: 10	0.566	0.429	0.834
D2V	dim: 500, algo: dbow, ws: 10, epoch: 20	0.565	0.427	0.833
D2V	dim: 500, algo: dbow, ws: 3, epoch: 20	0.564	0.427	0.832
KW	idf: none, tf: log, post: postnone	0.563	0.427	0.826
KW	idf: lin910, tf: sqrt_ratio, post: postnone	0.558	0.422	0.823
KW	idf: none, tf: sqrt_ratio, post: postnone	0.514	0.388	0.759
KW	idf: none, tf: log, post: postnone	0.474	0.360	0.694
RF	**base: tfidf, dim: 100, ntrees: 500, min_split: 10**	**0.394**	**0.297**	**0.584**
RF	base: tfidf, dim: 100, ntrees: 100, min_split: 10	0.394	0.297	0.584
RF	base: tfidf, dim: 500, ntrees: 100, min_split: 10	0.388	0.293	0.575
RF	base: tfidf, dim: 500, ntrees: 500, min_split: 10	0.388	0.292	0.575
RF	base: bow_unary, dim: 100, ntrees: 100, min_split: 10	0.343	0.259	0.507

Tabulka 5.2: Nejlepší klasifikátory podle F-measure – veřejné zakázky

met	parametry metody	F-meas	prec	recall
KW	**idf: lin910, tf: sqrt_ratio, post: postnone**	**0.414**	**0.295**	**0.698**
KW	idf: lin910, tf: over0.005, post: post1000	0.411	0.292	0.693
D2V	**dim: 100, algo: dbow, ws: 10, epoch: 10**	**0.396**	**0.280**	**0.677**
D2V	dim: 100, algo: dbow, ws: 10, epoch: 5	0.394	0.279	0.667
KW	idf: sharp910, tf: log, post: post1000	0.393	0.279	0.667
KW	idf: sharp910, tf: sqrt_ratio, post: post1000	0.393	0.279	0.667
KW	idf: sharp910, tf: over0.005, post: post1000	0.393	0.279	0.667
D2V	dim: 100, algo: dbow, ws: 1, epoch: 3	0.390	0.277	0.661
D2V	dim: 100, algo: dbow, ws: 3, epoch: 10	0.390	0.276	0.665
D2V	dim: 100, algo: dbow, ws: 3, epoch: 3	0.390	0.277	0.659
FT	**dim: 500, epoch: 20, ws: 3**	**0.378**	**0.269**	**0.633**
FT	dim: 100, epoch: 20, ws: 3	0.377	0.268	0.633
FT	dim: 100, epoch: 20, ws: 10	0.376	0.268	0.631
FT	dim: 500, epoch: 20, ws: 10	0.376	0.268	0.631
FT	dim: 500, epoch: 20, ws: 1	0.376	0.268	0.630
RF	**base: tfidf, dim: 100, ntrees: 100, min_split: 10**	**0.114**	**0.083**	**0.182**
RF	base: tfidf, dim: 100, ntrees: 500, min_split: 10	0.107	0.078	0.171
RF	base: tfidf, dim: 500, ntrees: 100, min_split: 10	0.104	0.076	0.165
RF	base: tfidf, dim: 500, ntrees: 500, min_split: 10	0.099	0.072	0.158
RF	base: tfidf, dim: 100, ntrees: 500, min_split: 100	0.097	0.071	0.152

Tabulka 5.3: Nejlepší klasifikátory podle F-measure – smlouvy

klasifikátor	parametry	průměr AUC
KW	idf: lin910, tf: log, post: postnone	0.962
KW	idf: lin910, tf: sqrt_ratio, post: postnone	0.962
FT	dim: 100, epoch: 20, ws: 3	0.958
FT	dim: 100, epoch: 20, ws: 10	0.958
FT	dim: 100, epoch: 20, ws: 1	0.958
FT	dim: 500, epoch: 20, ws: 10	0.957
FT	dim: 500, epoch: 20, ws: 1	0.957
D2V	dim: 500, algo: dbow, ws: 1, epoch: 10	0.950
D2V	dim: 500, algo: dbow, ws: 10, epoch: 10	0.950
D2V	dim: 500, algo: dbow, ws: 3, epoch: 10	0.949
D2V	dim: 500, algo: dbow, ws: 10, epoch: 20	0.948
D2V	dim: 500, algo: dbow, ws: 3, epoch: 20	0.947
KW	idf: none, tf: log, post: postnone	0.945
KW	idf: none, tf: sqrt_ratio, post: postnone	0.944
KW	idf: lin910, tf: sqrt_ratio, post: post1000	0.840
RF	base: tfidf, dim: 500, ntrees: 100, min_split: 10	0.632
RF	base: tfidf, dim: 500, ntrees: 500, min_split: 10	0.632
RF	base: tfidf, dim: 100, ntrees: 100, min_split: 10	0.630
RF	base: tfidf, dim: 100, ntrees: 500, min_split: 10	0.629
RF	base: bow_unary, dim: 500, ntrees: 500, min_split: 10	0.592

Tabulka 5.4: AUC podle klasifikátorů – veřejné zakázky

lepších modelů založených na jiných metodách. V případě zakázek činí tento rozdíl 8 procentních bodů, v případě smluv dokonce 26. Pro zakázky byly nejlepší klasifikátory metody FastText, avšak podobnou úspěšnost zaznamenal i nejlepší klasifikátor založený na klíčových slovech; ostatní klasifikátory byly horší téměř o 3 procentní body. V případě smluv se však do popředí dostala naopak metoda klíčových slov a Doc2Vec; je však třeba zmínit, že absolutní hodnoty všech tří metrik jsou u smluv horší než u zakázek.

5.5.3 Srovnání klasifikátorů podle AUC

Posuzování klasifikátorů podle metriky AUC je obtížnější, protože tato metrika má samostatnou hodnotu pro každou kategorii. AUC jsme měřili pouze při testování veřejných zakázek; použití pro smlouvy by dávalo smysl až v momentě, kdy by pro každou kategorii bylo netriviálně velké množství označených smluv náležících do dané kategorie.

V tabulce 5.4 jsou k dispozici nejlepší modely pro každou metodu (stejně jako v předešlé sekci) s průměrnými hodnotami AUC přes všechny kategorie. Klasifikátory random forests jsou opět zásadně horší než ostatní modely. U ostatních klasifikátorů – s výjimkou jednoho modelu využívajícího klíčová slova – se průměrné AUC pohybuje od 94 do 96 procent. Lepší výsledky mají některé modely klíčových slov a FastText.

Agregace pomocí průměru je však relativně hrubá; AUC některých kategorií se může zásadně lišit podle toho, který klasifikátor použijeme. Tuto skutečnost dokládá tabulka 5.5, kde uvádíme hodnoty AUC pro nejlepší klasifikátor metody podle F-measure. Vypsali jsme 30 kategorií s nejvyšším rozptylem AUC podle metody; v tomto výběru přibližně v polovině případů má lepší AUC metoda klíčových slov, v druhé polovině metoda FastText.

kategorie	FT	D2V	KW
doprava-posta	0.9759	**0.9984**	0.8831
stav-generic	**0.9793**	0.8795	0.9555
stav-prace	**0.9857**	0.8975	0.9577
stav-technik	0.9149	0.8963	**0.9777**
social-vzdelavani	0.9176	0.9838	**0.9894**
social-knihovny	0.9062	0.9509	**0.9722**
kancelar-generic	**0.9517**	0.8869	0.9320
doprava-dily	0.9105	0.9568	**0.9725**
stav-inspekce	**0.9521**	0.8907	0.9179
stav-materialy	**0.8751**	0.8156	0.8566
stroje-generic	**0.9624**	0.9037	0.9304
kancelar-spotrebice	0.8623	0.9041	**0.9175**
stav-instal	0.8438	0.8457	**0.8934**
stav-inzenyr	**0.9340**	0.8788	0.9063
stav-montaz	**0.9052**	0.8540	0.8904
social-zdravotni	0.9291	**0.9800**	0.9476
stroje-elektricke	**0.9196**	0.8754	0.9169
remeslo-textil	0.9148	0.8932	**0.9415**
kancelar-nabytek	**0.9548**	0.9079	0.9387
kancelar-cisteni	0.9406	0.9784	**0.9798**
techsluzby-generic	**0.9177**	0.8780	0.9153
stroje-prumysl	**0.9503**	0.9082	0.9346
stav-konstr	0.9220	0.8878	**0.9265**
legal-reality	0.8768	0.8755	**0.9115**
social-kultura	0.9181	**0.9577**	0.9462
it-generic	**0.9830**	0.9454	0.9712
jine-opravy	0.9243	0.8937	**0.9286**
doprava-generic	**0.9675**	0.9310	0.9583
stav-dokonceni	**0.9430**	0.9058	0.9292
jidlo-voda	0.7845	0.8096	**0.8200**

Tabulka 5.5: AUC kategorií s největším rozptylem hodnot – veřejné zakázky

5.5.4 Srovnání kategorií podle AUC

Na základě hodnot AUC je také možné srovnat obtížnost rozpoznání a řazení jednotlivých kategorií. Tabulka 5.6 představuje průměrné hodnoty AUC kategorií z nejlepších klasifikátorů. Z tabulky vyplývá, že nelze říci, že by některé oblasti byly snadnější k rozpoznání než jiné. Rovněž obecně neplatí, že by generické kategorie byly snazší nebo obtížnější k rozpoznání, než jiné. Mezi kategorie s nejvyšším AUC patří například `zdrav-leciva`, o které jsme se zmiňovali v sekci 3.1.2.

5.6 Diskuze

V této kapitole jsme otestovali čtyři různé metody klasifikace, které jsme navrhli při analýze možných řešení (kapitola 4). Metody klasifikace jsme trénovali s různě nastavenými parametry na trénovací sadě veřejných zakázek; testy jsme prováděli na veřejných zakázkách a ručně označené sadě smluv.

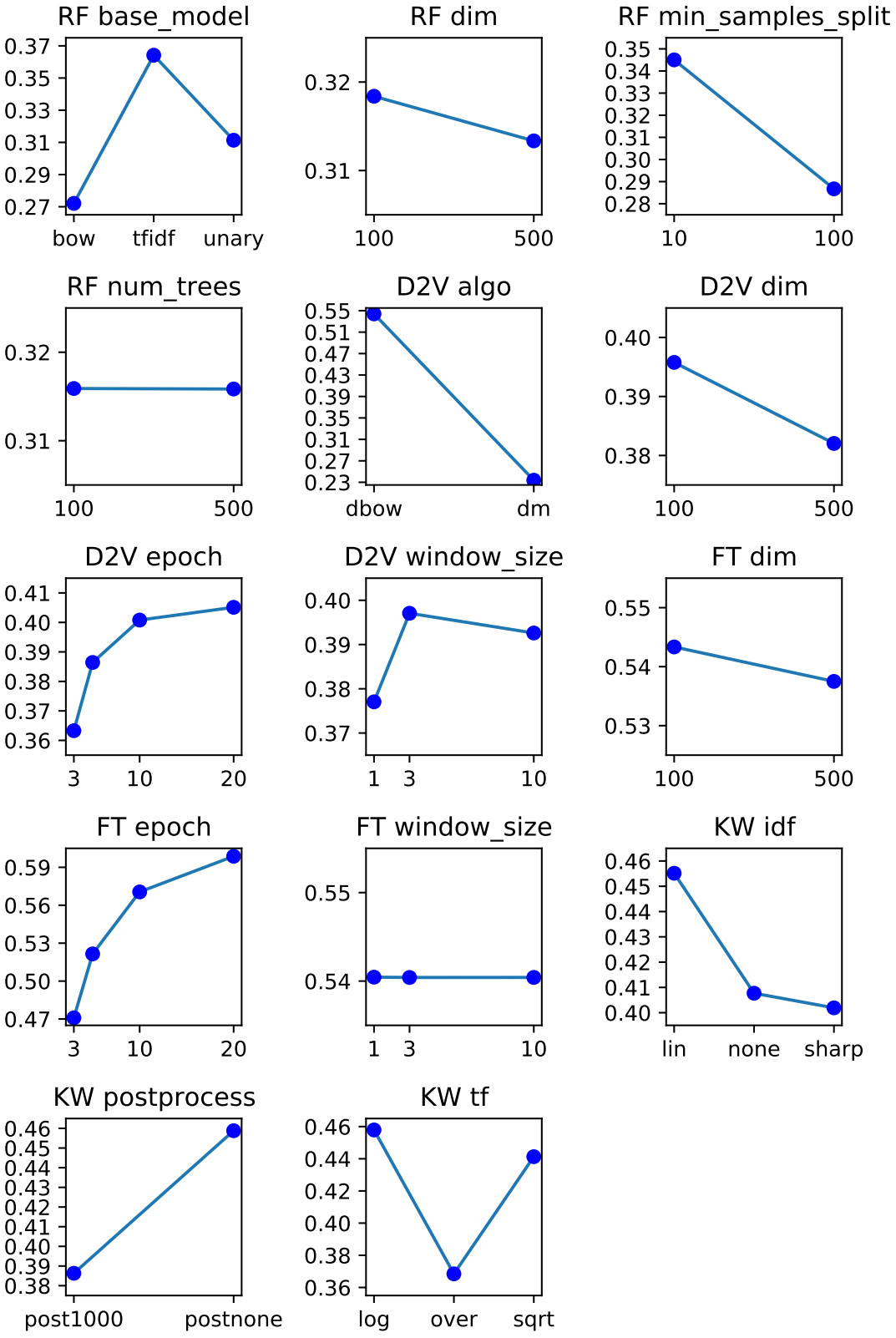
Výsledky testování jednoznačně ukazují, že metoda random forests klasifikuje smlouvy i zakázky mnohem méně úspěšněji než ostatní metody. Úspěšnost klasifikátorů Doc2Vec, FastText a klíčových slov záleží na metrice, podle které modely posuzujeme, a datové sadě. Například podle metriky F-measure je pro veřejné zakázky nejúspěšnější klasifikátor metody FastText, avšak pro smlouvy vykázal lepší výsledky klasifikátor založený na klíčových slovech. Metrika AUC jako nejlepší označila také klasifikátor metody klíčových slov, avšak pouze při agregaci hodnot; při porovnávání hodnot pro samostatné kategorie vykázala v polovině případů lepší výsledky metoda FastText.

Porovnání výsledků podle parametrů ukázalo, že pro FastText i Doc2Vec je důležitým parametrem počet epoch (iterací trénování), ačkoliv pro Doc2Vec vyšší hodnota snižuje úspěšnost při klasifikaci smluv. Pro Doc2Vec byla pro obě datové sady jednoznačně úspěšnější struktura sítě DBOW. F-measure metody random forests nejvíce závisí na struktuře vektorů dokumentů; nejlepší klasifikaci přineslo použití TF-IDF. Metoda klíčových slov přinesla, pokud jde o vliv parametrů na kategorizaci dokumentů, nejednoznačné výsledky.

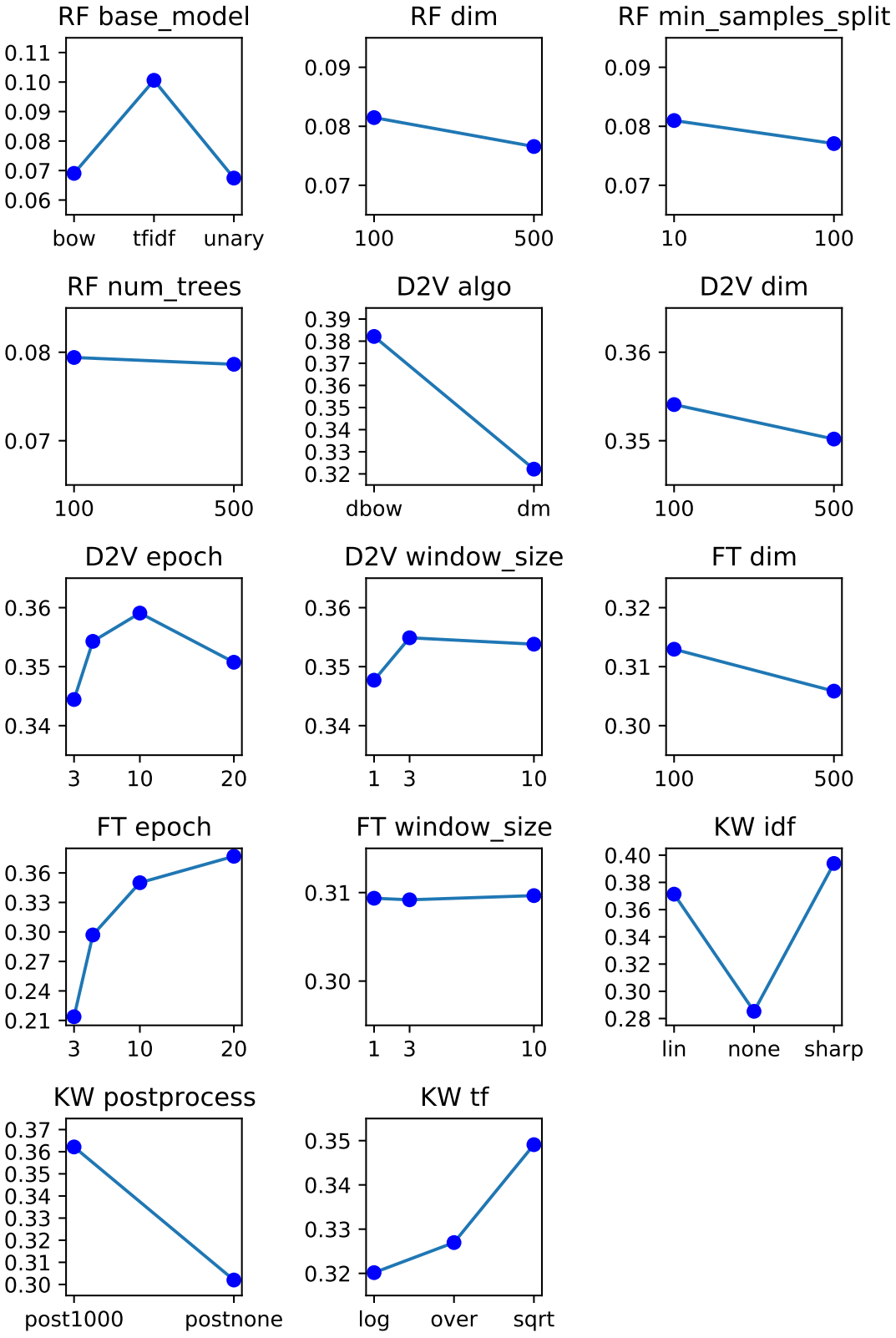
Ve spolupráci s autorem portálu Hlídač státu jsme jako řešení vybrali metodu klíčových slov. Její výsledky jsou srovnatelné s metodami založenými na slovních vektorech, avšak vybraný klasifikátor umožňuje snadné rozšiřování kategorií a pro autora je snadno pochopitelný. Nevýhodou tohoto řešení je nutnost spravovat seznamy n-gramů pro jednotlivé kategorie. Ukazuje se však, že velké množství n-gramů je duplicitní (nachází se ve více kategoriích) a je možné s nimi pracovat dohromady; zároveň klasifikátor funguje i při relativně nízkém množství slov pro každou kategorii.

kategorie	auc	kategorie	auc
agro-les	0.9710	doprava-koleje	0.8596
energie-elektrina	0.9691	**social-generic**	**0.8593**
telco-sluzby	0.9561	doprava-special	0.8592
energie-jina	0.9505	zdrav-opravy	0.8591
energie-paliva	0.9450	prirodnizdroj-tezba	0.8590
zdrav-leciva	0.9401	it-system	0.8587
stav-zeleznice	0.9368	doprava-opravy	0.8580
zdrav-generic	**0.9333**	kancelar-bezpecnost	0.8563
doprava-pohotovost	0.9314	social-vzdelavani	0.8538
stav-prace	0.9309	it-sw	0.8535
agro-generic	**0.9260**	**legal-generic**	**0.8535**
social-pece	0.9251	it-servery	0.8532
doprava-udrzba	0.9240	finance-ucetni	0.8526
stav-generic	**0.9228**	jidlo-potrava	0.8525
stav-silnice	0.9205	doprava-silnice	0.8523
energie-generic	**0.9196**	telco-tv	0.8503
zdrav-pristroje	0.9196	prirodnizdroj-pisky	0.8492
finance-sluzby	0.9146	energie-sluzby	0.8476
doprava-nakladni	0.9122	finance-poradenstvi	0.8469
it-generic	**0.9106**	kancelar-nabytek	0.8439
stav-mosty	0.9081	**kancelar-generic**	**0.8391**
kancelar-stroje	0.9076	jine-admin	0.8384
stav-voda	0.9070	it-opravy	0.8384
it-hw	0.9018	**vyzkum-generic**	**0.8379**
doprava-osobni	0.8968	stav-konstr	0.8374
it-vyvoj	0.8925	telco-site	0.8369
stroje-laborator	0.8903	stav-vedeni	0.8363
agro-tezba	0.8896	social-zdravotni	0.8355
kancelar-cisteni	0.8882	it-site	0.8355
doprava-sluzby	0.8849	doprava-posta	0.8342
kancelar-tisk	0.8840	stroje-prumysl	0.8341
doprava-lidi	0.8834	**jidlo-generic**	**0.8298**
telco-generic	**0.8834**	remeslo-hudba	0.8264
finance-generic	**0.8827**	**remeslo-generic**	**0.8263**
legal-pravni	0.8823	doprava-dily	0.8245
doprava-generic	**0.8822**	social-kultura	0.8241
techsluzby-odpady	0.8791	social-knihovny	0.8237
jine-pohostinstvi	0.8777	stav-dokonceni	0.8234
jine-jidelny	0.8752	stav-technik	0.8192
prirodnizdroj-chemie	0.8746	stav-inzenyr	0.8186
social-skoleni	0.8720	techsluzby-uklid	0.8172
kancelar-nabor	0.8703	jine-opravy	0.8136
it-sw-sluzby	0.8689	**bezpecnost-generic**	**0.8111**
agro-zahrada	0.8688	**techsluzby-generic**	**0.8106**
jine-pruzkum	0.8683	remeslo-textil	0.8083
stroje-generic	**0.8667**	stroje-elektricke	0.7950
prirodnizdroj-generic	**0.8637**	kancelar-spotrebice	0.7895
remeslo-odevy	0.8626	legal-reality	0.7864
stav-inspekce	0.8624	stav-montaz	0.7806
techsluzby-cisteni	0.8614	stav-instal	0.7579
zdrav-osobni	0.8606	stav-materialy	0.7513
marketing-generic	**0.8600**	jidlo-voda	0.7254

Tabulka 5.6: AUC podle kategorií – veřejné zakázky



Obrázek 5.1: Výsledky F-measure podle použitých parametrů – veřejné zakázky



Obrázek 5.2: Výsledky F-measure podle použitých parametrů – smlouvy

6. Implementace řešení

Tato kapitola se zabývá praktickou implementací a nasazením funkčního řešení klasifikace textů na portálu Hlídač státu. Vzhledem k tomu, že systém klasifikace je vysoce nezávislý na zdrojovém kódu portálu (což byl i předpoklad v zadání, viz sekce 2.3), rozhodli jsme se jej distribuovat jako samostatný dockerový modul. V následujících sekcích si vysvětlíme výhody i nevýhody tohoto řešení, dále si také popíšeme API, kterým zdrojový kód Hlídače komunikuje s naším modulem.

Vzhledem k tomu, že pro implementaci experimentů jsme využili jazyk Python, jsme jej chtěli využít i ve finálním modulu. Toto rozhodnutí se však neobešlo bez problémů, které zmíníme v závěru kapitoly.

Detailnější popis zdrojových kódů je možno nalézt v příloze C. Sekce C.6 obsahuje podrobný návod, jak spustit dockerový modul.

6.1 Architektura řešení

Architektura řešení a vztah naší implementace ke zdrojovému kódu webu Hlídače státu byla do jisté míry určena již v zadání (viz sekce 2.3): mělo jít o nezávislý modul, k němuž budou ostatní služby přistupovat pomocí jasně definovaného API. I tak je vhodné se zamyslet nad výhodami a nevýhodami tohoto přístupu a případných alternativách.

Výhodou nezávislosti modulu je možnost volného výběru vhodných technologií. Vzhledem k tomu, že experimenty s jednotlivými modely strojového učení jsme prováděli pomocí jazyku Python a knihoven podporujících tento jazyk, dává smysl, že chceme Python využít i v finální implementaci. Backendové řešení webu Hlídače je však z velké části založeno na technologiích .NET a frameworku ASP.NET. Pokud bychom vyžadovali řešení úzce integrované s backendem, musel by náš klasifikátor být založen na .NET a knihovnách, které jej podporují (ačkoliv bychom měli určitou volnost ve výběru programovacího jazyka). Ze sémantického hlediska navíc naše řešení skutečně je na webu Hlídače nezávislé, protože jej lze smysluplně použít i mimo tento web. Jasně viditelné je to u části řešení, jejímž úkolem je lemmatizovat text. Další výhoda je spíše „pedagogická“: v případě API méně závislého modulu je obvykle třeba toto API lépe definovat, což typicky přináší zlepšení jeho kvality.

Nevýhodou nezávislého modulu je určitá ztráta výkonu: přenos dat je komplikovanější než v případě úzce propojených modulů. Skutečná ztráta výkonu se pochopitelně může lišit podle toho, kde vzájemně nezávislé moduly běží (na stejném stroji, ve stejné síti, v různých sítích) a může být snížená například kešováním výsledků. V případě našeho modulu však časové nároky na přenos dat nejsou kritické: za prvé, klasifikace textu se neprovádí v reálném čase, za druhé, samotné zpracování textu (lemmatizace) trvá řádově déle, než přenos dat.

Další nevýhodou je větší technologická složitost propojení nezávislých modulů. Je třeba vybrat vhodný *middleware*, implementaci komunikační vrstvy. Tímto problémem se budeme zabývat v následující sekci.

6.2 Výběr middlewaru

Ačkoliv termín *middleware* má více různých významů, nejčastěji se používá definice z oboru distribuovaného softwaru. Podle Schantze a Shmidta [57] je middleware *vrstva mezi aplikací a hardwarovou nebo softwarovou infrastrukturou, vytvořená za účelem snadnějšího a efektivnějšího vývoje znovupoužitelných aplikací.* V našem případě hledáme takovou mezivrstvu, která umožní komunikaci mezi naší implementací klasifikace textu a webem Hlídače státu. Naše požadavky na middleware jsou následující:

- Middleware může být bezestavový: klasifikace textu nebude udržovat vnitřní stav pro jednotlivé spojení nebo klienty.
- Protože se klasifikace textu může používat i mimo web Hlídače, musí být služby middlewaru široce podporované napříč knihovnami programovacích jazyků.
- API pro klasifikaci textu bude jednoduché: v podstatě lze popsat tak, že vstupem bude specifikace smlouvy a výstupem seznam kategorií. Proto upřednostňujeme middleware s menší složitostí a menším rozsahem funkčnosti.
- Vzhledem k tomu, že smlouvy jsou uložené ve formátu JSON, by middleware měl tento formát podporovat.

Vzhledem k uvedeným požadavkům by bylo nevhodné používat middleware založený na RPC (například *gRPC*[1]), které nevyhovuje požadavkům na jednoduchost a bezestavovost, navíc také nativně nepodporuje formát JSON. Podobné problémy má protokol SOAP (přenos zpráv je omezený na formát XML).

Výhodnějším postupem je zapojení protokolu HTTP a architektonického vzoru REST (*Representational state transfer*), představeného Royem Fieldingem v jeho disertační práci [18]. Samotný protokol HTTP byl navržen pro přístup k prostředkům (*resources*) na síti World Wide Web, identifikovatelnými pomocí URL. REST představuje generalizaci tohoto konceptu, kdy podobně přístupné mohou být jiné služby mimo WWW. Mezi principy převzaté z HTTP patří:

- architektura klient-server (jako protipól architektury peer-to-peer),
- bezestavovost spojení,
- podpora kešování odpovědí,
- jednotné rozhraní pro služby (nezávislé na konkrétní implementaci služby),
- identifikace prostředků pomocí URL.

Využití HTTP zároveň znamená, že dojde i k převzetí jeho konkrétnějších vlastností:

- formátu zprávy (hlavičky a tělo požadavku nebo odpovědi),

[1] `https://grpc.io/`

- významu hlaviček, například
  - `Content-Type` pro označení MIME typu zprávy v těle požadavku,
  - hlaviček ovlivňující funkci cache, `Cache-Control`, `Expires` a další,
- metod (`GET`, `POST`, `HEAD` atd.) a jejich sémantiky,
- stavových kódů odpovědí.

Jako slabé stránky architektury REST je někdy uváděno to, co je ve skutečnosti její zamýšlenou vlastností; často se jedná o bezestavovost. (*It's not a bug, it's a feature.*) Objektivnější problém REST se však týká snahy o obecnější využití již zavedeného protokolu. HTTP pracuje s prostředky pomocí sémantických metod: `GET` znamená získat prostředek, `POST` označuje nahrání nové verze prostředku, `DELETE` smazání prostředku. Ne každé zamýšlené API lze však do této sémantiky pohodlně převést: databázové operace se reprezentují snadno (`POST /fotografie /budova_matfyzu.jpeg`, `GET /fotografie/budova_matfyzu.jpeg` atd.), avšak operace volání procedur (bližší řešením RPC) se převádí hůře. Na webu lze rovněž nalézt dlouhé diskuze například o rozdílu mezi metodami `PUT` a `POST`, což poukazuje na to, že sémantická stránka REST není perfektně vyřešena.

Přes zmíněné nedostatky jsme se rozhodli v řešení použít právě architekturu REST a protokol HTTP.

6.3 Virtualizace, běhové prostředí a deployment

Naprostá většina netriviálního softwaru obsahuje závislosti na běhovém prostředí, knihovnách a ostatních programech o určité verzi nebo rozsahu verzí. I naše řešení je závislé na externích knihovnách jazyku Python, stejně jako na specifické verzi Pythonu. Problém zajištění všech nutných závislostí (zvaný také *dependency hell*) také lehce souvisí s nasazením produkční verze softwaru (*deployment*), jemuž se také budeme věnovat.

Naše požadavky na vyřešení závislostí a deploymentu modulu pro klasifikaci textu vychází z faktu, že si nejsme dopředu jisti, v jakém prostředí vlastně modul poběží. Virtualizované servery Hlídače státu obvykle používají platformu Windows, ale kvůli některým operacím jsou k dispozici i linuxové stroje – není však striktně specifikované, která distribuce se používá. Ačkoliv je samotný Python multiplatformní jazyk, pro provoz modulu budeme vyžadovat unixové prostředí – důvody vysvětlíme v sekci 6.4.4. Následuje seznam možných způsobů řešení:

- *Ruční instalace* všech závislých knihoven nepřipadá v úvahu, protože ta se liší podle použité platformy.
- Využití pythonvského virtuálního prostředí – *virtualenv*. Na stroji, kde poběží modul, je vytvořen oddělený repozitář knihoven Pythonu. Do něj je možné nainstalovat knihovny jakékoliv verze podle potřeb softwaru. Nevýhodou je, že takto není „virtualizován“ samotný interpret Pythonu, který musí být nainstalován přímo v systému (ve správné verzi). Výhodou je, že samotný proces modulu pro klasifikaci není nijak virtualizován, v podstatě má pouze upravené proměnné prostředí.

- Využití *správců balíčků v Linuxu* (`apt`, `zypper` atd.) by nás omezilo na jednu konkrétní distribuci Linuxu.
- *Virtuální stroj* (plná virtualizace) by dovoloval použít jakékoliv prostředí pro běh modulu. Závislosti by byly nainstalovány přímo ve virtuálním stroji. Pro naše účely je však toto řešení komplikované a relativně náročné na systémové prostředky.
- *Virtualizace na úrovni operačního systému* (běh v kontejneru) je obdoba předcházejícího řešení. Místo virtualizace hardwaru je však virtualizované pouze prostředí, v němž je proces modulu spuštěný (systém souborů a IO), modul jinak běží jako proces jádra hostitelského systému. Výhodou je relativní nenáročnost a výpočetní efektivita.

Rozhodli jsme se použít řešení založené na virtualizaci na úrovni OS, konkrétně platformu Docker, která je v současnosti velkým trendem v oblasti cloud computingu [38; 67]. Využitím Dockeru se zároveň řeší otázka nasazení softwaru: obraz kontejneru ve veřejném repozitáři (zdaleka nejpoužívanějším je Docker Hub) se dá stáhnout a spustit zavoláním jediného příkazu.

Navzdory popularitě Dockeru musíme zdůraznit i nevýhody, které jeho použití přináší. Ačkoliv procesy, které jsou součástí běžícího kontejneru, jsou spuštěny přímo na jádru hostitelského operačního systému, přístup k nim je méně přímý, než u procesů, které v kontejneru neběží; ladění procesů je proto obtížnější. Obrazy v Docker Hubu sice jsou snadno přístupné, nemusí však vždy být bezpečné: mohou obsahovat bezpečnostní chyby, zvláště v případě, že autor obrazu jej dále nespravuje.

Kvůli problémům s laděním zdrojových kódů běžících v dockerovém kontejneru jsme se rozhodli podporovat i spuštění kódu pomocí *virtualenv*.

6.4 Popis modulu

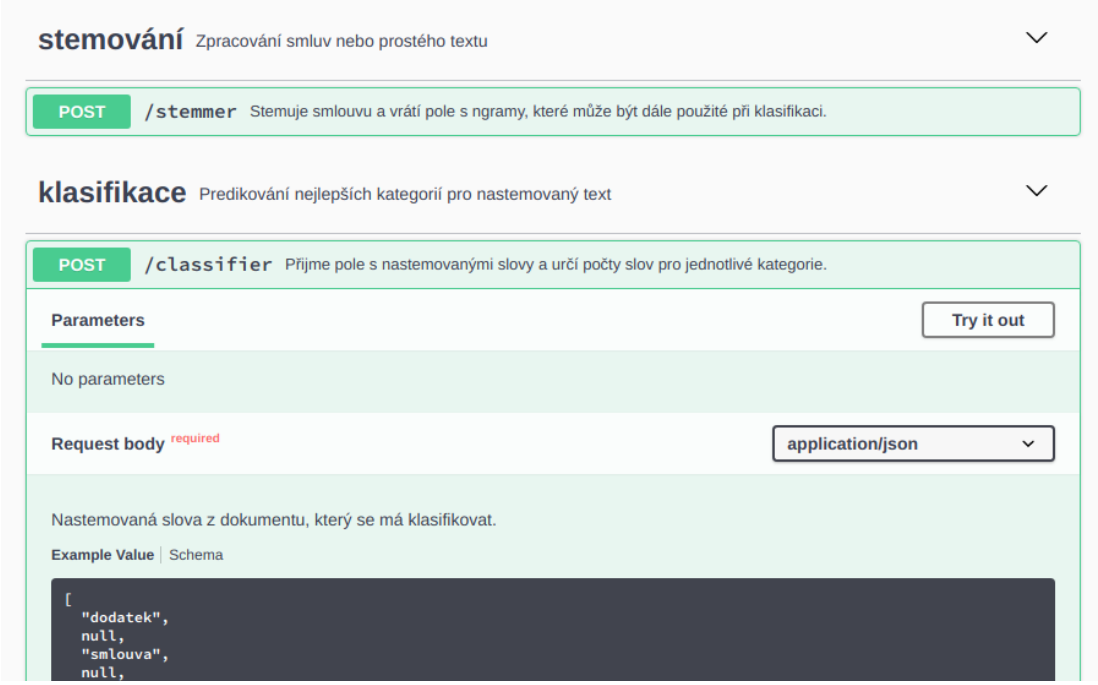
V této sekci popíšeme obsah a veřejné API obrazu Docker kontejneru pro klasifikaci textů.

Výchozím obrazem (na němž je založené sestavování našeho obrazu) je oficiální obraz pro příslušnou verzi jazyka Python, `python:3.7`. Do tohoto obrazu jsou doinstalovány potřebné externí knihovny a zkopírován zdrojový kód modulu pro klasifikaci textů, včetně potřebného modelu pro lematizér. Do obrazu jsou rovněž zkopírovány soubory s klíčovými slovy, které využívá námi vybraný model (viz sekce 5.6). Ve vývojových verzích obrazu, kdy byly testované i modely s řádově vyššími nároky na paměť (gigabyty), se jejich zdrojové soubory nepřikládaly do obrazu, ale připojily se externě při spouštění kontejneru[2] To však u současného modelu není nutné.

6.4.1 Popis REST API

Veřejné API obrazu se skládá ze tří koncových bodů – lemmatizéru, klasifikátoru a finalizátoru.

[2]Jde o funkci *volumes*, jejíž použití je doporučené pro velká a persistentní úložiště. Viz `https://docs.docker.com/storage/volumes/`.



Obrázek 6.1: Specifikace REST API: webový interface

Lemmatizér (`/stemmer`[3]) přijímá na vstupu JSON strukturu smlouvy z registru zakázek a na výstup vrátí pole lemmatizovaných slov. V poli se rovněž nacházejí hodnoty `null`, které označují místa s vyřazenými slovy – n-gramy se tvoří pouze z těch n-tic slov, které nejsou oddělena hodnotou `null`. Lemmatizace je časově nejnáročnější část klasifikace textu. Proto je uživatelům doporučeno, aby lemmatizovaná slova ukládali do cache.

Klasifikátor (`/classifier`) přijme výstup lemmatizéru a vrátí JSON objekt, kde klíče jsou názvy kategorií a hodnoty jsou pravděpodobnosti náležení dokumentu do příslušné kategorie, tak, jak je vrátil používaný model. Tento objekt je třeba předat finalizéru (`/finalizer`), který kategorie a hodnoty vyfiltruje, přepočítá a vrátí finální výběr kategorií a jejich pravděpodobností (opět jako objekt). Důvodem pro oddělení klasifikátoru a finalizéru je možnost ladění a diagnostiky „hrubého“ výstupu z modelu klasifikace.

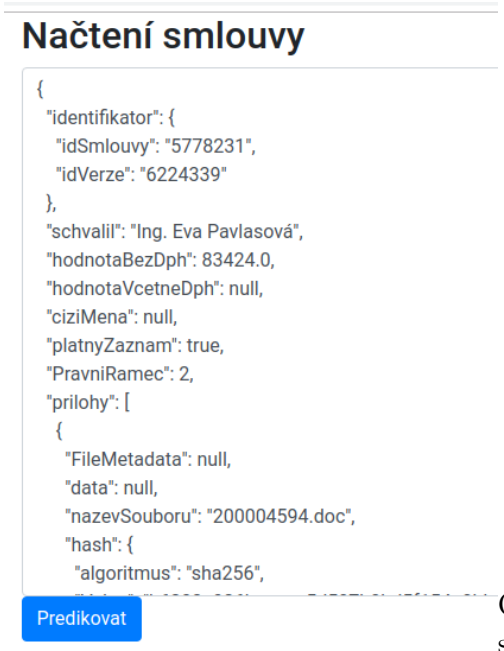
6.4.2 Specifikace REST API

Pro podrobnější specifikaci funkce API a jednotlivých endpointů jsme použili sadu nástrojů Swagger[4]. Swagger pracuje se specifikací REST API ve formátu OpenAPI[5]. Specifikace v tomto formátu umožňuje snadno vytvořit HTML dokumentaci API, z níž je možno i testovat vstup a výstup jednotlivých koncových bodů (obrázek 6.1). Také je možné generovat klientské moduly v různých programovacích jazycích.

[3]Původně jsme pro zpracování textu používali stemmer, po nasazení lemmatizéru jsme však název ponechali kvůli zpětné kompatibilitě.

[4]`https://swagger.io/`

[5]`https://swagger.io/resources/open-api/`



Obrázek 6.2: `/explain`: načtení smlouvy ve formátu JSON

Obrázek 6.3: `/explain`: seznam klíčových slov pro kategorie

6.4.3 Ladění modelu

Pro ladění modelu pro klasifikaci je možné použít koncový bod `/explain`. Tento bod funguje jako dynamická HTML stránka, zamýšlená k interaktivnímu použití, proto jej neuvádíme v REST API. Po vložení smlouvy v JSON formátu do textového pole a klepnutí na tlačítko *Predikovat* se zobrazí výsledek klasifikace a pro každou kategorii seznam nalezených klíčových slov. Rozhraní je zobrazené na obrázcích 6.2 a 6.3.

6.4.4 Poznámky k implementaci

Implementace modelů strojového učení a experimentů již byla popsána v předchozích kapitolách, proto se zmíníme pouze o těch částech zdrojového kódu, které se týkají REST API a HTTP serveru. Dotkneme se také otázky škálování.

Pro vytvoření API jsme využili *Flask*, webový framework pro jazyk Python.[6] a jeho modul *Flask-RESTful*. Na rozdíl od frameworků jako například *Django* poskytuje *Flask* nízkoúrovňovější služby, proto je vhodnější pro specifikaci REST endpointů. Endpoint `/explain` s dynamickým HTML výstupem používá šablonovací engine *Jinja2*.

Webové frameworky v Pythonu typicky podporují protokol *WSGI (Web Server Gateway Interface)*, skrze který s nimi komunikují HTTP servery. V našem případě byl jako HTTP server vybrán *uWSGI*, jehož výhodou je hlavně dobrá konfigurovatelnost.

[6] `https://www.fullstackpython.com/flask.html`

Škálování modulu

Při provozování modulu klasifikace smluv jsme museli řešit otázku výkonu a škálování. Celý modul je bezestavový a je využita virtualizace pomocí kontejnerů; mohlo by se tedy zdát, že škálovatelnosti přímočaře docílíme spuštěním více kontejnerů. Tento postup sice je funkční, ale má vyšší paměťové nároky; každá instance modulu (spuštěný kontejner) potřebuje načíst do paměti minimálně modul lemmatizéru, což tvoří řádově stovky megabytů. Další komplikace také přináší fakt, že každá instance používá samostatný HTTP server; pro transparentní použití API tedy bude třeba vytvořit proxy server, který bude instancím požadavky přeposílat.[7]

Pokusili jsme se proto zajistit škálovatelnost přímo na úrovni kontejneru a zajistit, aby HTTP server běžel ve více vláknech. Multithreading je však u Pythonu prakticky vyloučen; používaný interpreter *CPython* dovoluje v jednom procesu běh pouze jednoho vlákna, které vykonává příkazy interpreteru. Zámek vláken se označuje jako *Global Interpreter Lock (GIL)* a je součástí původní architektury. Důvodem jeho použití bylo zjednodušení správy paměti a importu nízkoúrovňových knihoven v jazyce C, které často nebyly thread-safe. [6] Souběžný průběh více vláken je možný pouze v případě, že vlákna provádí časově náročné akce, při kterých neprovádí příkazy Pythonu; příkladem může být čtení a zápis dat. To však není případem modulu pro klasifikaci textů. Lemmatizace sice dokáže běžet vícevláknově, klasifikace ale již ne, protože je napsaná přímo v Pythonu.

Kvůli popsaným komplikacím se jako náhrada multithreadingu v Pythonu často využívá multiprocessing, běh výpočtů ve více procesech se samostatnými oblasti v paměti. Úspory paměti je dosaženo tak, že samotný zdrojový kód, použité externí knihovny a další potřebná data jsou nahrány do paměti ještě v momentě, kdy běží jen jeden proces. Z tohoto procesu jsou pak spuštěny další procesy. Na unixových operačních systémech se však při vytváření nového procesu (pomocí funkce `fork`) paměť fyzicky nekopíruje, místo toho se virtuální paměť procesu odkáže na ty samé paměťové rámce; kopírování proběhne až v momentě, kdy jeden z procesů mění obsah bloku paměti.

V případě, že chceme využít tuto optimalizaci, jsme omezeni platformou na unixové systémy. V praxi se však také ukázalo, že je téměř nemožné zajistit, aby Python neměnil sdílené pamětové bloky. Ačkoliv se zdá, že sdílená data využíváme pouze pro čtení, Python je přesto mění. Hlavním důvodem je opět správa paměti, kdy Python buď mění hodnotu počítadla referencí datové struktury, případně je přesouvá v rámci provádění garbage collection. Důsledkem je pomalý nárůst množství použité paměti.

Nejpodrobnější popis příčin tohoto problému poskytli vývojáři služby Instagram, který rovněž používá HTTP servery založené na Pythonu. [36] Testovali například vypnutí výchozího garbage collectoru; i přes všechny optimalizace ale nedokázali zastavit nárůst spotřeby paměti. Závěr článku naznačuje, že procesy serveru se pravidelně restartují, což zabraňuje příliš velkému nárůstu spotřeby paměti; toto řešení jsme se, i přes zjevnou neeleganci, rozhodli použít.

[7] Vhodným způsobem by bylo použít Docker Compose, nástroj pro definování a spouštění systémů skládajících se z většího množství kontejnerů.
Viz `https://docs.docker.com/compose/`.

Závěr

Cílem této práce bylo analyzovat data o smlouvách a veřejných zakázkách a navrhnout řešení, které umožní smlouvy rozdělit do kategorií podle jejich textu. Na základě zkoumání datových sad jsme nalezli postup pro předzpracování textu a vybrali čtyři různé klasifikátory pro určení kategorií: random forests, word embeddings (dvě různá řešení) a klíčová slova. Určili jsme nejvhodnější metriky pro testování výkonnosti klasifikátorů a provedli experimenty na testovacích datových sadách. Zde se ukázalo, že metoda random forests má o mnoho nižší výkonnost než zbylé klasifikátory. Pro nasazení na produkčním serveru jsme vybrali metodu klíčových slov, protože nabízí nejlepší flexibilitu: je možné snadno upravovat i hotový klasifikátor.

Výstupem práce je implementace celého postupu pro kategorizaci smluv, připravená pro provoz v produkčním prostředí. Funguje jako samostatný modul, který s dalšími částmi webu Hlídač státu komunikuje pomocí HTTP a architektonického vzoru REST. Použití platformy Docker zajišťuje jednoduchost nasazení.

Implementace je v této chvíli skutečně nasazena na serverech portálu Hlídač státu a používá se pro klasifikaci smluv. Funkcionalita je zatím provozována v beta verzi: pro každou smlouvu je predikce kategorií viditelná v jejím detailu, avšak zatím pouze po přihlášení uživatele (registrovat se může kdokoliv). Také je možné vyhledávat podle vybrané kategorie. Po prvotním nasazení klasifikátoru došlo k sloučení některých kategorií (obzvlášť v oblasti stavitelství), ale také došlo k vytvoření kategorií nových.

Ve výzkumu klasifikace smluv a textů obecně se nabízí několik možností pro další výzkum. Bylo by užitečné otestovat na našich datových sadách i jiné klasifikátory, obzvláště různé další varianty neuronových sítí, které jsme nepoužili. Vzhledem k tomu, že se v našich experimentech nevyskytl jednoznačně nejlepší model, nabízí se také možnost experimentovat s jejich různými kombinacemi. Dalším možným postupem by bylo využít faktu, že texty smluv často vychází z veřejných zakázek, a pokusit se propojit konkrétní zakázky s konkrétními smlouvami.

Seznam použité literatury

[1] (2014). Projekt datových schránek byl podle NKÚ předražený, strmý růst nákladů se zastavil až letos. http://www.egov.cz/clanky/projekt-datovych-schranek-byl-podle-nku-predrazeny-strmy-rust-nakladu-se-zastavil-az-letos.

[2] (2015). Zákon č. 340/2015 Sb. o zvláštních podmínkách účinnosti některých smluv, uveřejňování těchto smluv a o registru smluv.

[3] Abdi, H. a Williams, L. J. (2010). Principal component analysis. *WIREs Computational Statistics*, **2**(4), 433–459. doi: 10.1002/wics.101. URL https://onlinelibrary.wiley.com/doi/abs/10.1002/wics.101.

[4] Alshuqayran, N., Ali, N. a Evans, R. (2016). A systematic mapping study in microservice architecture. In *2016 IEEE 9th International Conference on Service-Oriented Computing and Applications (SOCA)*, pages 44–51. doi: 10.1109/SOCA.2016.15.

[5] Andhavarapu, A. (2017). *Learning Elasticsearch*. Packt Publishing. ISBN 9781787129917. URL https://books.google.cz/books?id=2nc5DwAAQBAJ.

[6] Beazley, D. (2010). Understanding the python GIL. In *PyCON Python Conference. Atlanta, Georgia*.

[7] Bengio, Y., Ducharme, R., Vincent, P. a Jauvin, C. (2003). A neural probabilistic language model. *Journal of machine learning research*, **3**(Feb), 1137–1155.

[8] Bláha, M. O mně. https://www.michalblaha.cz/o-mne/.

[9] Bláha, M. (2016). O registru smluv. https://www.hlidacstatu.cz/texty/o-registru/.

[10] Bláha, M. (2016). O serveru. https://www.hlidacstatu.cz/texty/o-serveru/.

[11] Bláha, M. (2017). Hlídač státu: rok první. https://www.hlidacstatu.cz/texty/registr-smluv-rok-prvni/.

[12] Bojanowski, P., Grave, E., Joulin, A. a Mikolov, T. (2016). Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, **5**. doi: 10.1162/tacl_a_00051.

[13] Boser, B. E., Guyon, I. M. a Vapnik, V. N. (1992). A training algorithm for optimal margin classifiers. In *Proceedings of the Fifth Annual Workshop on Computational Learning Theory*, COLT '92, page 144–152, New York, NY, USA, 1992. Association for Computing Machinery. ISBN 089791497X. doi: 10.1145/130385.130401. URL https://doi.org/10.1145/130385.130401.

[14] Budějovický Budvar, n. p. (2017). Tisková produkce bigboardů. URL https://www.hlidacstatu.cz/Detail/2776962.

[15] Chandrashekar, G. a Sahin, F. (2014). A survey on feature selection methods. *Computers & Electrical Engineering*, **40**(1), 16–28.

[16] Davis, M. a Dürst, M. (2001). Unicode normalization forms.

[17] Ferri, C., Hernández-Orallo, J. a Modroiu, R. (2009). An experimental comparison of performance measures for classification. *Pattern Recognition Letters*, **30**(1), 27 – 38. ISSN 0167-8655. doi: https://doi.org/10.1016/j.patrec.2008.08.010. URL http://www.sciencedirect.com/science/article/pii/S0167865508002687.

[18] Fielding, R. T. a Taylor, R. N. (2000). *Architectural styles and the design of network-based software architectures*, volume 7. University of California, Irvine Doctoral dissertation.

[19] Fürnkranz, J. (1998). A study using n-gram features for text categorization.

[20] Ganatra, A. a Prajapati, P. (2012). A survey and current research challenges in multi-label classification methods. *International Journal of Soft Computing and Engineering (IJSCE)*, **2**.

[21] Grira, N., Crucianu, M. a Boujemaa, N. (2005). Unsupervised and semi-supervised clustering: a brief survey. *A Review of Machine Learning Techniques for Processing Multimedia Content*.

[22] Hladká, B. a Holub, M. (2019). Intro to decision trees and random forests. https://ufal.mff.cuni.cz/~holub/2019/docs/lec.DT-RF.basic.2019-10-16.pdf.

[23] Homolce, N. N. (2019). Rámcová dohoda na dodávky cytostatik pro specializovanou péči (antineoplastika). URL https://www.hlidacstatu.cz/VerejneZakazky/Zakazka/05876103F263B00E5E67FDB739884743.

[24] Ibrahim, R., Elbagoury, A., Kamel, M. S. a Karray, F. (2017). Tools and approaches for topic detection from Twitter streams: survey. *Knowledge and Information Systems*. doi: 10.1007/s10115-017-1081-x.

[25] Jeong, C. (2007). *Fundamental of Development Administration*. Scholar. ISBN 9789833813094. URL https://books.google.cz/books?id=SoJRAQAACAAJ.

[26] Joulin, A., Grave, E., Bojanowski, P. a Mikolov, T. (2016). Bag of tricks for efficient text classification. *CoRR*, **abs/1607.01759**. URL http://arxiv.org/abs/1607.01759.

[27] Kocan, B. (2004). Průvodce veřejnými zakázkami v Evropské unii. *Praha: Czech Trade*.

[28] Kondratyuk, D., Gavenciak, T., Straka, M. a Hajic, J. (2018). Lemmatag: Jointly tagging and lemmatizing for morphologically-rich languages with brnns. *CoRR*, **abs/1808.03703**. URL http://arxiv.org/abs/1808.03703.

[29] Kononenko, O., Baysal, O., Holmes, R. a Godfrey, M. W. (2014). Mining modern repositories with elasticsearch. In *Proceedings of the 11th Working Conference on Mining Software Repositories*, pages 328–331. ACM.

[30] Kowsari, K., Meimandi, K. J., Heidarysafa, M., Mendu, S., Barnes, L. E. a Brown, D. E. (2019). Text classification algorithms: A survey. *CoRR*, **abs/1904.08067**. URL http://arxiv.org/abs/1904.08067.

[31] Krajská zdravotní, a.s. (2019). Dodávka léčivých přípravků s účinnou látkou NIVOLUMAB II. URL https://www.hlidacstatu.cz/VerejneZakazky/Zakazka/537A545BBADFA27DF304ADECD8147E35.

[32] KruV - RS (Kraj Vysočina) (2016). Poskytnutí dotace v rámci GP Rozvoj podnikatelů 2016. URL https://www.hlidacstatu.cz/Detail/504553.

[33] Kuo, F. Y. a Sloan, I. H. (2005). Lifting the curse of dimensionality. *Notices of the AMS*, **52**(11), 1320–1328.

[34] Le, Q. V. a Mikolov, T. (2014). Distributed representations of sentences and documents. *CoRR*, **abs/1405.4053**. URL http://arxiv.org/abs/1405.4053.

[35] Lebret, R. a Lebret, R. (2013). Word emdeddings through Hellinger PCA. *CoRR*, **abs/1312.5542**. URL http://arxiv.org/abs/1312.5542.

[36] Li, Z. (2017). Copy-on-write friendly Python garbage collection. https://instagram-engineering.com/copy-on-write-friendly-python-garbage-collection-ad6ed5233ddf.

[37] Manning, C., Raghavan, P. a Schütze, H. (2008). *Introduction to Information Retrieval*. Cambridge University Press. ISBN 9781139472104. URL https://books.google.cz/books?id=t1PoSh4uwVcC.

[38] Merkel, D. (2014). Docker: lightweight Linux containers for consistent development and deployment. *Linux journal*, **2014**(239), 2.

[39] Mikolov, T., Chen, K., Corrado, G. S. a Dean, J. (2013). Efficient estimation of word representations in vector space. *CoRR*, **abs/1301.3781**.

[40] Mikolov, T., Sutskever, I., Chen, K., Corrado, G. a Dean, J. (2013). Distributed representations of words and phrases and their compositionality. In *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'13, page 3111–3119, Red Hook, NY, USA, 2013. Curran Associates Inc.

[41] Ministerstvo obrany (2013). Oblek pro potápěče. URL https://www.hlidacstatu.cz/VerejneZakazky/Zakazka/652F4D7A06C1D60CE7B9F7DBC4E36087.

[42] Ministerstvo obrany (2019). Run-flaty pro vojenská vozidla. URL https://www.hlidacstatu.cz/VerejneZakazky/Zakazka/41DB366C7D50E004E5D66DF04ED01324.

[43] Ministerstvo vnitra (2010). Velký Beranov 356 – RD. URL https://www.hlidacstatu.cz/Detail/pre279140329.

[44] Ministerstvo vnitra, odbor eGovernmentu (2019). Metodický návod k aplikaci zákona o registru smluv. https://www.mvcr.cz/soubor/metodicky-navod-k-aplikaci-zakona-o-registru-smluv-jez-slouzi-k-zakladni-orientaci-v-problematice-a-prinasi-zakladni-odpovedi-na-casto-kladene-dotazy.aspx.

[45] Mitchell, T. M. (1997). *Machine Learning*. McGraw-Hill, Inc., New York, NY, USA, 1 edition. ISBN 0070428077, 9780070428072.

[46] Město Tábor (2018). Lávka pro pěší a cyklisty a chodníky v prostoru Černých mostů v Táboře. URL https://www.hlidacstatu.cz/VerejneZakazky/Zakazka/53FA9AB31028CAEA965E77BC7D579E77.

[47] Město Český Těšín (2016). Dodatek č. 7 ke Smlouvě o umístění technického zařízení na sběr. URL https://www.hlidacstatu.cz/Detail/53705.

[48] Novák, P. Cena za datovou zprávu opět klesne, stát ušetří miliony korun. https://www.mvcr.cz/clanek/cena-za-datovou-zpravu-opet-klesne-stat-usetri-miliony-korun.aspx.

[49] Obec Halenkov (2016). Revitalizace zeleně Halenkov – zelená zeleň. URL https://www.hlidacstatu.cz/VerejneZakazky/Zakazka/71F33CFDE534D5CFF22C06C8467ABEAB.

[50] Onderka, J. (2015). Nové metody zpracování textu pro klasifikaci emocí. Master's thesis, Vysoké učení technické v Brně. Fakulta elektrotechniky a komunikačních technologií, Brno.

[51] Organizace spojených národů, E. a. s. r. (2018). *UN E-Government Survey 2018*. ISBN 9789211232080. URL https://publicadministration.un.org/egovkb/en-us/Reports/UN-E-Government-Survey-2018.

[52] Pan, S. J. a Yang, Q. (2010). A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, **22**(10), 1345–1359. doi: 10.1109/TKDE.2009.191.

[53] Papadimitriou, C. H., Raghavan, P., Tamaki, H. a Vempala, S. (2000). Latent semantic indexing: A probabilistic analysis. *Journal of Computer and System Sciences*, **61**(2), 217 – 235. ISSN 0022-0000. doi: https://doi.org/10.1006/jcss.2000.1711. URL http://www.sciencedirect.com/science/article/pii/S0022000000917112.

[54] Patra, A. a Singh, D. (2013). A survey report on text classification with different term weighing methods and comparison between classification algorithms. *International Journal of Computer Applications*, **75**, 14–18. doi: 10.5120/13122-0472.

[55] Reed, R. D. a Marks, R. J. (1998). *Neural Smithing: Supervised Learning in Feedforward Artificial Neural Networks*. MIT Press, Cambridge, MA, USA. ISBN 0262181908.

[56] Řehůřek, R. a Sojka, P. (2010). Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, pages 45–50, Valletta, Malta, May 2010. ELRA. http://is.muni.cz/publication/884893/en.

[57] Schantz, R. a Schmidt, D. (2007). *Middleware for Distributed Systems*. ISBN 9780470050118. doi: 10.1002/9780470050118.ecse241.

[58] Sperandei, S. (2014). Understanding logistic regression analysis. *Biochemia medica*, **24**, 12–8. doi: 10.11613/BM.2014.003.

[59] Štědroň, B. (2007). *Úvod do eGovernmentu v České republice: právní a technickỳ průvodce*. Úřad vlády České republiky. ISBN 9788087041253. URL https://books.google.cz/books?id=PoWSIQAACAAJ.

[60] Straka, M., Hajic, J. a Straková, J. (2016). UDPipe: trainable pipeline for processing conll-u files performing tokenization, morphological analysis, pos tagging and parsing. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*, pages 4290–4297.

[61] Straková, J., Straka, M. a Hajič, J. (2014). Open-Source Tools for Morphology, Lemmatization, POS Tagging and Named Entity Recognition. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 13–18, Baltimore, Maryland, June 2014. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/P/P14/P14-5003.pdf.

[62] Státní zemědělský intervenční fond (2018). Dohoda o poskytnutí dotace z Programu rozvoje venkova ČR. URL https://www.hlidacstatu.cz/Detail/4909460.

[63] Vrabec, P. a Bláha, M. (2019). Stát jako vrchnost končí, říká internetový byznysmen Blaha. https://neovlivni.cz/stat-jako-vrchnost-konci-rika-internetovy-byznysmen-blaha/.

[64] Vysoká škola báňská – Technická univerzita Ostrava (2018). Contract on the use of the high performance computing cluster. URL https://www.hlidacstatu.cz/Detail/53705.

[65] Vysoká škola ekonomická v Praze (2017). Rekonstrukce gastronomického provozu kuchyně v Italské budově - areál VŠE v

Praze. URL https://www.hlidacstatu.cz/VerejneZakazky/Zakazka/F8DAE46150AA8CBAF67394C8DD256F7D.

[66] Wu, X. a Zhou, Z. (2016). A unified view of multi-label performance measures. *CoRR*, **abs/1609.00288**. URL http://arxiv.org/abs/1609.00288.

[67] Yadav, A. K., Garg, M. a kol. (2019). Docker containers versus virtual machine-based virtualization. In *Emerging Technologies in Data Mining and Information Security*, pages 141–150. Springer.

[68] Zhang, T. (2004). Solving large scale linear prediction problems using stochastic gradient descent algorithms. In *Proceedings of the Twenty-First International Conference on Machine Learning*, ICML '04, page 116, New York, NY, USA, 2004. Association for Computing Machinery. ISBN 1581138385. doi: 10.1145/1015330.1015332. URL https://doi.org/10.1145/1015330.1015332.

[69] Šípoš, G., Spáč, S. a Kollárik, M. (2015). Not in force until published online: What the radical transparency regime of public contracts achieved in Slovakia.

A. Příloha: Popis kategorií

V této kapitole jsou popsány jednotlivé oblasti a jejich kategorie. Ke každému záznamu je uveden krátký popis, odpovídající prefixy CPV kódů a příklady veřejných zakázek s touto kategorií. V elektronické PDF verzi se u CPV kódů a příkladů nachází příslušné odkazy do portálu Hlídač státu.

Identifikátor kategorie, psaný `proporcionálním písmem`, občas neodpovídá přesně popisu kategorie. Je to z toho důvodu, že identifikátory byly přiřazené kategoriím již při obdržení seznamu, ačkoliv se pro některé kategorie později ukázalo, že neodpovídají skutečnému obsahu kategorie. Protože se ale identifikátory nyní používají i ve výsledcích, rozhodl jsem se je zde uvést nezměněné.

Celkem máme 20 oblastí a 104 kategorií (počítáno včetně generických).

A.1 Informační technologie

`it-generic`: Obecná kategorie pro informační technologie

- CPV 302: *Počítače*
- CPV 48: *Balíky programů a informační systémy*
- CPV 503: *Opravy a údržba osobních počítačů, kancelářského, telekomunikačního a audiovizuálního zařízení a související služby*
- CPV 72: *Informační technologie: poradenství, vývoj programového vybavení, internet a podpora*

`it-hw`: Nákup, prodej a správa hardwaru

- CPV 302: *Počítače*
- Příklad: 04C16: *Počítačové sestavy, notebooky a monitory 2014-2018 – 011/2015*

`it-sw`: Nákup softwaru

- CPV 48: *Balíky programů a informační systémy*
- Příklad: B2FDD: *Nákup Office 2016 Standard a Professional Plus*

`it-servery`: Informační systémy a servery a jejich správa

- CPV 488: *Informační systémy a servery*
- Příklad: FB816: *Portace IDP serverů*

`it-opravy`: Opravy a údržba výpočetní techniky

- CPV 503: *Opravy a údržba osobních počítačů, kancelářského, telekomunikačního a audiovizuálního zařízení a související služby*
- Příklad: 0D06C: *Kompletní správa výpočetní techniky a komunikační infrastruktury*

`it-vyvoj`: Poradenství a vývoj programového vybavení

- CPV 72: *Informační technologie: poradenství, vývoj programového vybavení, internet a podpora*
- Příklad: 8B495: *SW pro podporu řízení kolektivního výzkumu Klastru OMNIPACK*
- Příklad: 47BCA: *Zajištění technické podpory produktů Oracle Magistrátu města Brna*

`it-system`: Analýza informačních systémů a poradenské služby

- CPV 7222: *Systémové a technické poradenské služby*
- CPV 7224: *Analýza systémů a programovací služby*
- CPV 7225: *Systémové a podpůrné služby*
- CPV 726: *Výpočetní podpora a poradenské služby*
- Příklad: 39610: *Zajištění provozu a rozvoje IS SZIF*

`it-sw-sluzby`: Softwarové služby; komplexní vývoj, analýza dat atd.

- CPV 7223: *Vývoj programového vybavení na zakázku*
- CPV 7226: *Služby programového vybavení*
- CPV 723: *Datové služby*
- CPV 725: *Výpočetní služby*
- CPV 728: *Audity a testování počítačů*
- CPV 729: *Zálohování počítačů a zpracování počítačových katalogů*
- Příklad: BB56F: *Služby certifikační autority pro Finanční správu České republiky*

`it-site`: Internetové služby a počítačové sítě

- CPV 724: *Internetové služby*
- CPV 727: *Počítačové sítě*
- Příklad: 046D6: *Poskytování WAN VPN pro ŘSD – Datová síť*

A.2 Stavebnictví

`stav-generic`: Obecná kategorie pro stavebnictví

- CPV 44: *Stavební konstrukce a materiály; pomocné výrobky pro konstrukce (mimo elektrické přístroje)*
- CPV 45: *Stavební práce*
- CPV 71: *Architektonické, stavební, technické a inspekční služby*
- CPV 75123: *Administrativa bytové výstavby*
- CPV 34946: *Konstrukční materiály železničních tratí a dodávky pro stavbu železničních tratí*
- CPV 351131: *Bezpečnostní zařízení pro staveniště*
- CPV 4331: *Stroje používané ve stavebnictví*
- CPV 433: *Stavební stroje a zařízení*
- CPV 436: *Části strojů pro doly, lomy a stavebnictví*
- CPV 507: *Opravy a údržba technických stavebních zařízení*
- CPV 51541: *Instalace a montáž důlních, těžebních, stavebních a metalurgických strojů*
- CPV 7011: *Výstavba nemovitého majetku*
- CPV 79993: *Správa staveb a zařízení*
- CPV 909112: *Stavební úklid*
- CPV 507: *Opravy a údržba technických stavebních zařízení*
- CPV 51: *Instalační a montážní služby (mimo programového vybavení)*
- CPV 71: *Architektonické, stavební, technické a inspekční služby*

`stav-materialy`: Stavební konstrukce a materiály

- CPV 44: *Stavební konstrukce a materiály; pomocné výrobky pro konstrukce (mimo elektrické přístroje)*
- Příklad: DEAA8: *Asfaltová emulze pro SÚS PK, p.o. (rok 2014)*

`stav-prace`: Stavební práce

- CPV 45: *Stavební práce*
- Příklad: 074C9: *Stavební úpravy rodinného domu Srbská 522/2, Liberec 11*

`stav-konstr`: Stavební konstrukce – mosty, tunely, odpočívadla

- CPV 4522: *Konstrukční a stavební práce*

- Příklad: E276E: *Rekonstrukce mostu ev.c. 26836-20 Nové Zákupy*

`stav-mosty`: Stavební úpravy mostů a tunelů

- CPV 45221: *Stavební úpravy mostů a tunelů, šachet a podchodů*
- Příklad: 291D6: *Malý Labský náhon – rekonstrukce mostu*

`stav-vedeni`: Stavební práce pro potrubní, telekomunikační a elektrické vedení

- CPV 45231: *Stavební práce pro potrubní, telekomunikační a elektrické vedení*
- Příklad: E110B: *Rekonstrukce vodovodu DN 500 ul. Mariánskohorská*

`stav-silnice`: Výstavba, zakládání a povrchové práce pro silnice

- CPV 45233: *Výstavba, zakládání a povrchové práce pro komunikace*
- Příklad: 87542: *Rekonstrukce komunikace v ulici Hradební Prachatice*

`stav-zeleznice`: Stavební úpravy pro železnici

- CPV 45234: *Stavební úpravy pro železniční a lanové dopravní systémy*
- Příklad: DAE2D: *Odstranění propadu rychlosti na trati Krnov – Šumperk*

`stav-voda`: Výstavba vodních děl (nádrží, jezů), protipovodňová opatření

- CPV 4524: *Výstavba vodních děl*
- Příklad: 386D6: *Rekonstrukce vodní nádrže Okrouhlík*

`stav-montaz`: Stavební montážní práce; montáž elektroinstalace, pokládka kabelů, izolace, větrání, klimatizace, sanitární zařízení

- CPV 453: *Stavební montážní práce*
- Příklad: D3244: *Stavební úpravy budovy rektorátu ČZU 2014*

`stav-dokonceni`: Práce při dokončování budov; omítací práce, montáž oken a dveří, malířské práce, pokládání podlah

- CPV 454: *Práce při dokončování budov*
- Příklad: 7C19E: *Dostavba a nové využití objektu "Střelnice"Liberec*

`stav-technik`: Opravy a údržba technických stavebních zařízení

- CPV 507: *Opravy a údržba technických stavebních zařízení*
- Příklad: E6464: *Oprava (rekonstrukce) oběžného výtahu páternoster*

`stav-instal`: Instalační a montážní služby elektrického vybavení (motorů, transformátorů atd.), měřících a kontrolních přístrojů, komunikačních zařízení

- CPV 51: *Instalační a montážní služby (mimo programového vybavení)*

- Příklad: 1C8E0: *Slavnostní nasvětlení hradu a přístupové cesty SH Bouzov*

`stav-inspekce`: Architektonické, stavební, technické a inspekční služby

- CPV 71: *Architektonické, stavební, technické a inspekční služby*
- Příklad: 8972C: *I/44 Postřelmov - Velké Losiny - zpracování GP a realizace MPV*

`stav-inzenyr`: Technicko-inženýrské služby, poradenství, projektování

- CPV 713: *Technicko-inženýrské služby*
- Příklad: A6C28: *Libeňský most, Praha 7 a 8, č. akce 999984 - Správce stavby a koordinátor*

A.3 Doprava

`doprava-generic`: Obecná kategorie pro dopravu

- CPV 34: *Přepravní zařízení a pomocné výrobky pro přepravu*
- CPV 60: *Přepravní služby (mimo přepravu odpadů)*
- CPV 63: *Pomocné a doplňkové dopravní služby; provozování cestovních agentur*
- CPV 09132: *Automobilový benzin*
- CPV 091342: *Nafta*
- CPV 0913423: *Bionafta*
- CPV 09211: *Mazací oleje a maziva*
- CPV 501: *Opravy a údržba vozidel a příslušenství k nim a související služby*
- CPV 502: *Opravy a údržba letadel, železničního a silničního vozového parku a lodí a související služby*
- CPV 5114: *Instalace a montáž motorů*
- CPV 641: *Poštovní a kurýrní služby*

`doprava-osobni`: Motorová vozidla pro přepravu osob

- CPV 3411: *Osobní vozidla*
- Příklad: 39520: *Dodávka užitkového osobního automobilu N1*

`doprava-special`: Specializovaná vozidla pro přepravu osob, typicky pohotovostní vozidla

- CPV 34114: *Specializovaná vozidla*
- Příklad: F3361: *Vozidla - Moderní technika a technologie Policie ČR*

`doprava-lidi`: Motorová vozidla pro přepravu více než 10 lidí

- CPV 3412: *Motorová vozidla pro přepravu více než 10 lidí*
- Příklad: 67F19: *Dodávka městského autobusu v roce 2013*

`doprava-nakladni`: Nákladní vozy (pro přepravu zboží, odvoz odpadu, hasičská vozidla atd.)

- CPV 3413: *Motorová vozidla pro přepravu zboží*
- CPV 3414: *Motorová vozidla pro velké zatížení*
- Příklad: B4A11: *Dodávka nákladních vozidel N2 – 6 ks*

`doprava-udrzba`: Vozidla silniční údržby (sypače, zametače, kropící vozy)

- CPV 341444: *Vozidla silniční údržby*
- Příklad: 81D6C: *Strojové vybavení na úklid zpevněných cest za účelem snížení prašnosti*

`doprava-pohotovost`: Nákladní vozidla pro pohotovostní služby

- CPV 341442: *Vozidla pro pohotovostní služby*
- Příklad: 14D38: *Nákup dopravního automobilu pro jednotku Sboru dobrovolných hasičů Pzeň*

`doprava-dily`: Díly a příslušenství k motorovým vozidlům

- CPV 343: *Díly a příslušenství k motorovým vozidlům a motorům*
- Příklad: C1CBB: *Dodávky originálních náhradních dílů pro služební dopravní prostředky*

`doprava-koleje`: Železniční a tramvajové lokomotivy a vozidla

- CPV 346: *Železniční a tramvajové lokomotivy a kolejová vozidla a jejich díly*
- Příklad: A3324: *Odkup 50 železničních vozů od ÖBB*

`doprava-silnice`: Silniční zařízení (zařízení pro údržbu, značení, protihlukové stěny atd.)

- CPV 3492: *Silniční zařízení*
- Příklad: 7F3C4: *Dodávka bezobslužného parkovacího systému pro Nemocnici Jihlava*

`doprava-opravy`: Opravy a údržba vozidel a příslušenství k nim a související služby

- CPV 501: *Opravy a údržba vozidel a příslušenství k nim a související služby*
- Příklad: 98844: *Vozidla kategorie M1, N1 – oprava a servis*

`doprava-sluzby`: Služby silniční dopravy (taxislužba, pravidelná přeprava, pronájem vozidel)

- CPV 601: *Služby silniční dopravy*
- Příklad: B0186: *Přechodné zabezpečení stanoveného rozsahu dopravní obslužnosti Libereckého kraje*

`doprava-posta`: Poštovní služby

- CPV 641: *Poštovní a kurýrní služby*
- Příklad: EB3DA: *Doručování Poštovních zásilek a Doporučených poštovních zásilek*

A.4 Strojírenství

`stroje-generic`: Obecná kategorie pro strojírenství

- CPV 16: *Zemědělské stroje*
- CPV 31: *Elektrické strojní zařízení, přístroje, zařízení a spotřební materiál, osvětlení*
- CPV 38: *Laboratorní, optické a přesné přístroje a zařízení (mimo skel)*
- CPV 42: *Průmyslové stroje*
- CPV 43: *Stroje pro hlubinné a povrchové dobývání a stavební stroje*
- CPV 505: *Opravy a údržba čerpadel, ventilů, kohoutů a kovových nádob a strojního zařízení*
- CPV 515: *Instalace a montáž strojů a přístrojů*

`stroje-elektricke`: Elektrické stroje

- CPV 31: *Elektrické strojní zařízení, přístroje, zařízení a spotřební materiál, osvětlení*
- Příklad: C069E: *Dodávka 2 kusů transformátorů 110/22kV o výkonu 63 MVA*

`stroje-laborator`: Laboratorní přístroje a zařízení

- CPV 38: *Laboratorní, optické a přesné přístroje a zařízení (mimo skel)*
- Příklad: 24527: *Dodávka zařízení na stlačený vzduch*

`stroje-prumysl`: Průmyslové stroje

- CPV 42: *Průmyslové stroje*
- CPV 43: *Stroje pro hlubinné a povrchové dobývání a stavební stroje*
- Příklad: 5197E: *Vysokozdvižné vozíky pro AČR*

A.5 Telekomunikace

`telco-generic`: Obecná kategorie pro telekomunikace

- CPV 32: *Rozhlas, televize, komunikace, telekomunikace a související zařízení*
- CPV 642: *Telekomunikační služby*
- CPV 5033: *Údržba telekomunikačních zařízení*
- CPV 513: *Instalace a montáž komunikačních zařízení*

`telco-tv`: Televize a rozhlas, zařízení pro záznam i přehrávání

- CPV 323: *Televizní a rozhlasové přijímače, zařízení pro nahrávání zvuku nebo videa nebo duplikační přístroje*
- Příklad: BA646: *Audiovizuální technika pro potřeby SU*

`telco-site`: Sítě, síťová zařízení a příslušenství (např. kabeláž)

- CPV 324: *Sítě*
- CPV 325: *Telekomunikační přístroje na přenos dat*
- Příklad: 1439D: *Vybavení odborných učeben výpočetní technikou*

`telco-sluzby`: Telekomunikační služby (připojení k internetu, hlasové služby, televizní a rozhlasové vysílání)

- CPV 642: *Telekomunikační služby*
- Příklad: 0C800: *Služby operátora mobilních zařízení*

A.6 Zdravotnictví

`zdrav-generic`: Obecná kategorie pro zdravotnictví

- CPV 33: *Zdravotnické přístroje, farmaceutika a prostředky pro osobní péči*
- CPV 504: *Opravy a údržba zdravotnických a přesných přístrojů*
- CPV 514: *Instalace a montáž zdravotnických vč chirurgických přístrojů*
- CPV 504: *Opravy a údržba zdravotnických a přesných přístrojů*

`zdrav-pristroje`: Zdravotnické přístroje

- CPV 331: *Zdravotnické přístroje*
- Příklad: 73DD0: *Operační stůl pro gynekologický operační sál*

`zdrav-leciva`: Léčiva

- CPV 336: *Léčivé přípravky a zdravotnické prostředky*

- Příklad: 33CA5: *Léčivé přípravky pro zažívací trakt a metabolismus A16A B04*

`zdrav-opravy`: Opravy a údržba zdravotnických přístrojů

- CPV 504: *Opravy a údržba zdravotnických a přesných přístrojů*
- Příklad: 4E811: *Servis RTG přístrojů*

`zdrav-osobni`: Prostředky pro osobní péči (kosmetika, parfémy, hygiena apod.)

- CPV 337: *Prostředky pro osobní péči*
- Příklad: FB6B1: *Toaletní papír, ubrousky, papírové utěrky a papírové ručníky*

A.7 Potravinářství

`jidlo-generic`: Obecná kategorie pro potravinářství

- CPV 03: *Produkty zemědělství, hospodářské produkty, produkty akvakultury, lesnictví a související produkty*
- CPV 15: *Potraviny, nápoje, tabák a související produkty*
- CPV 4111: *Pitná voda*

`jidlo-potrava`: Nákup a prodej potravin

- CPV 15: *Potraviny, nápoje, tabák a související produkty*
- CPV 033: *Hospodářské produkty, produkty myslivosti a akvakultury*
- Příklad: B4A98: *Rámcová smlouva na dodávky mléka a mléčných výrobků*

`jidlo-voda`: Pitná voda, nápoje alkoholické a nealkoholické

- CPV 159: *Nápoje, tabák a související produkty*
- CPV 4111: *Pitná voda*
- Příklad: A09EB: *Dodávka pitné vody v barelech včetně pronájmu výdejníků na vodu*
- Příklad: 39CE8: *Rozšíření technologie pro výrobu piva*

A.8 Bezpečnost

`bezpecnost-generic`: Bezpečnostní a armádní vybavení

- CPV 35: *Bezpečnostní, hasičské, policejní a ochranné vybavení*
- CPV 506: *Opravy a údržba bezpečnostních a obranných materiálů*
- CPV 5155: *Instalace a montáž zbraňových systémů*
- CPV 519: *Instalace a montáž naváděcích a kontrolních systémů*
- Příklad: 55212: *Dodávka střeliva 9 mm Luger*
- Příklad: 9FCF1: *Servis systémů technické ochrany vybraných objektů MERO ČR, a. s.*
- Příklad: AE9E4: *GŘ OL – dodávka vrchních oděvů stejnokroje*
- Příklad: B28ED: *Záchranářská nosítka s příslušenstvím*

A.9 Přírodní zdroje

`prirodnizdroj-generic`: Obecná kategorie pro přírodní zdroje

- CPV 14: *Produkty těžebního průmyslu, kovové suroviny a související produkty*
- CPV 24: *Chemické výrobky*
- CPV 41: *Shromážděná a upravená voda*

`prirodnizdroj-pisky`: Písky a jíly

- CPV 142: *Písky a jíly*
- Příklad: F129E: *Drcené kamenivo na posyp komunikací pro zimní období 2018-2019*

`prirodnizdroj-chemie`: Chemické výrobky

- CPV 24: *Chemické výrobky*
- Příklad: 13D3A: *Dodávka kyseliny sírové technické*

`prirodnizdroj-tezba`: Produkty těžebního průmyslu

- CPV 14: *Produkty těžebního průmyslu, kovové suroviny a související produkty*
- Příklad: 2C321: *051/3/2015 Hutní materiál 1/2019*
- Příklad: A4431: *Brusné materiály*

A.10 Energetika

`energie-generic`: Obecná kategorie pro energetiku

- CPV 09: *Ropné produkty, paliva, elektrická energie a ostatní zdroje energie*
- CPV 65: *Veřejné služby*
- CPV 3112: *Generátory*
- CPV 3113: *Alternátory*
- CPV 3114: *Chladící věže*
- CPV 45251: *Výstavba elektráren a tepláren*
- CPV 71314: *Energetické a související služby*
- CPV 713231: *Projektování energetických rozvodných sítí*

`energie-paliva`: Paliva, ropa a zemní plyn

- CPV 091: *Paliva*
- CPV 092: *Ropné a uhelné produkty a produkty z ropných frakcí*
- Příklad: F6915: *Město Kraslice – nákup zemního plynu*
- Příklad: 5D3D8: *Dodávky motorové nafty*

`energie-elektrina`: Elektrická energie

- CPV 0931: *Elektrická energie*
- Příklad: C1334: *Dodávky elektřiny pro Město Roudnice nad Labem*
- Příklad: 24BAA: *Výběr dodavatele elektrické energie*

`energie-jina`: Paliva, ropa, zemní plyn, elektřina

- CPV 09: *Ropné produkty, paliva, elektrická energie a ostatní zdroje energie*
- Příklad: 95385: *Zprostředkování obchodu s elektrickou energií na Českomoravské komoditní burze*

`energie-sluzby`: Veřejné služby pro energie

- CPV 65: *Veřejné služby*
- Příklad: 4D3A1: *Energetické služby a dodávky*
- Příklad: 45142: *Sdružené služby dodávek plynu a elektrické energie*

A.11 Zemědělství

`agro-generic`: Obecná kategorie pro zemědělství

- CPV 16: *Zemědělské stroje*
- CPV 77: *Zemědělské, lesnické, zahradnické služby a služby v oblasti akvakultury a včelařství*
- CPV 5152: *Instalace a montáž zemědělských a lesnických strojů*
- CPV 034: *Produkty lesnictví a těžby dřeva*
- CPV 772: *Služby v oblasti lesnictví*
- CPV 773: *Zahradnické služby*

`agro-les`: Lesnictví

- CPV 034: *Produkty lesnictví a těžby dřeva*
- CPV 0345: *Produkty lesních školek*
- CPV 772: *Služby v oblasti lesnictví*
- Příklad: 50CA8: *Dodávky materiálů pro pěstební práce včetně jejich aplikace*

`agro-tezba`: Těžba dřeva a související služby

- CPV 7721: *Služby v oblasti těžby dřeva*
- CPV 7722: *Impregnace dřeva*
- CPV 7723: *Služby související s lesnictvím*
- Příklad: 8702C: *Těžba a přibližování dřevní hmoty na OM na území KRNAP a jeho OP*

`agro-zahrada`: Zahradnické služby

- CPV 773: *Zahradnické služby*
- Příklad: 718E7: *Obnova zeleně městského parku v Rožnově pod Radhoštěm*

A.12 Kancelářské služby

`kancelar-generic`: Obecná kategorie pro kancelářské služby

- CPV 22: *Tiskařské výrobky a související produkty*
- CPV 30: *Kancelářské a počítací stroje, zařízení a potřeby mimo nábytek a balíky programů*
- CPV 39: *Nábytek (včetně kancelářského), zařízení interiéru, domácí spotřebiče (mimo osvětlení) a čisticí prostředky*

- CPV 795: *Pomocné kancelářské služby*
- CPV 796: *Nábor zaměstnanců*
- CPV 797: *Vyhledávací a bezpečnostní služby*
- CPV 798: *Tiskařské a související služby*
- CPV 799: *Různé provozní a s podnikáním související služby*
- CPV 503: *Opravy a údržba osobních počítačů, kancelářského, telekomunikačního a audiovizuálního zařízení a související služby*

`kancelar-tisk`: Tisk

- CPV 22: *Tiskařské výrobky a související produkty*
- CPV 798: *Tiskařské a související služby*
- Příklad: 88687: *Rámcová smlouva na nákup tiskovin*
- Příklad: 0D9D2: *Redakční činnost, výroba a distribuce periodika Plzeňský kraj*
- Příklad: 1ACBC: *Dodávka české odborné literatury*

`kancelar-stroje`: Kancelářské stroje (mimo počítače a nábytek)

- CPV 301: *Kancelářské strojní zařízení a potřeby mimo počítače, tiskárny a nábytek*
- Příklad: FABE9: *Tonery a cartridge 06/2016*

`kancelar-nabytek`: Nábytek, domácí spotřebiče a čistící prostředky

- CPV 39: *Nábytek (včetně kancelářského), zařízení interiéru, domácí spotřebiče (mimo osvětlení) a čisticí prostředky*
- Příklad: BE000: *Dodávka a restaurování nábytku, lavic a podlah a dodávka replik lustrů*

`kancelar-spotrebice`: Domácí spotřebiče

- CPV 397: *Domácí spotřebiče*
- Příklad: E3998: *Dodávka gastrovybavení pro projekt „Multifunkční velkoprostorové odborné učebny – gastrocentra≪“*

`kancelar-cisteni`: Čisticí výrobky

- CPV 398: *Čisticí a lešticí výrobky*
- Příklad: C1CD8: *Drogistické zboží 009-2013*

`kancelar-nabor`: Nábor zaměstnanců

- CPV 796: *Nábor zaměstnanců*

- Příklad: 0907E: *Externí lektoři na výuku v Rozvojových programech*

`kancelar-bezpecnost`: Vyhledávací a bezpečnostní služby, ostraha

- CPV 797: *Vyhledávací a bezpečnostní služby*
- Příklad: 44E62: *Zajištění ostrahy, úklidu a správy sídla Krajského pozemkového úřadu*

A.13 Řemeslná práce

`remeslo-generic`: Obecná kategorie pro řemeslnou práci

- CPV 18: *Oděvy, obuv, brašnářské výrobky a doplňky*
- CPV 19: *Usně a textilie, plastové a pryžové materiály*
- CPV 37: *Hudební nástroje, sportovní zboží, hry, hračky, materiál pro řemeslné a umělecké práce a příslušenství*

`remeslo-odevy`: Oděvy

- CPV 18: *Oděvy, obuv, brašnářské výrobky a doplňky*
- Příklad: 37641: *Obuv sportovní*
- Příklad: F77BB: *Kukly – čepice*

`remeslo-textil`: Textilie

- CPV 19: *Usně a textilie, plastové a pryžové materiály*
- Příklad: 729C4: *Dodávka textilií – Perlan*

`remeslo-hudba`: Hudba, sport, vybavení hřišť atd.

- CPV 37: *Hudební nástroje, sportovní zboží, hry, hračky, materiál pro řemeslné a umělecké práce a příslušenství*
- Příklad: 227F6: *Dětské hřiště v areálu Komunitního centra Maják II*
- Příklad: 34282: *Nová rolba na úpravu ledové plochy na ZS Krystal*
- Příklad: 2436F: *Výběrové řízení na dodavatele hudebních nástrojů*

A.14 Sociální služby, vzdělávání a rekreace

`social-generic`: Obecná kategorie pro sociální služby, vzdělávání a rekreaci

- CPV 80: *Vzdělávání a školení*
- CPV 85: *Zdravotní a sociální péče*
- CPV 92: *Rekreační, kulturní a sportovní služby*

- CPV 98: *Jiné služby pro veřejnost, sociální služby a služby jednotlivcům*

`social-vzdelavani`: Vzdělávání

- CPV 801: *Základní vzdělávání*
- CPV 802: *Středoškolské vzdělávání*
- CPV 803: *Vyšší vzdělávání*
- Příklad: 84AAB: *Vzdělávání pro pracovníky poskytovatelů sociálních služeb v Jihomoravském kraji*
- Příklad: 89A93: *Cyklus přednášek pro žáky středních škol v rámci projektu Podpora přírodovědného a technického vzdělávání*
- Příklad: 53106: *Zajištění vzdělávání zaměstnanců Ministerstva práce a sociálních věcí ČR, Odboru implementace fondů EU*

`social-skoleni`: Školení

- CPV 805: *Školení*
- CPV 804: *Vzdělávání dospělých a jiné vzdělávání*
- Příklad: 6EBCC: *Uspořádání školících a jazykových kurzů*
- Příklad: 9A419: *Odborným vzděláváním ke zvýšení efektivity zaměstnanců TEDOM a.s.*

`social-zdravotni`: Podpora zdravotní péče, ozdravné pobyty

- CPV 851: *Zdravotnické zabezpečení*
- Příklad: 31D4C: *Ozdravný pobyt dětí I. stupně základních škol*

`social-pece`: Sociální péče

- CPV 853: *Sociální péče a související služby*
- CPV 981: *Služby poskytované dobrovolnými organizacemi*
- Příklad: B095F: *Zajištění sociální služby, nízkoprahová zařízení pro děti a mládež v sociálně vyloučených romských lokalitách*
- Příklad: 57266: *Poskytování sociální služby – azylové domy v Pardubickém kraji*

`social-kultura`: Podpora filmu a rozhlasu

- CPV 921: *Služby v oblasti filmu a videa*
- CPV 922: *Rozhlasové a televizní služby*
- CPV 923: *Zábavní vysílání*

- Příklad: 701A0: *Propagační spoty Moravskoslezského kraje v rámci projektu Moravskoslezský kraj - kraj plný zážitků III*
- Příklad: 00141: *Výroba a vysílání televizního magazínu*

`social-knihovny`: Kulturní instituce (knihovny, archivy, muzea)

- CPV 925: *Knihovny, archivy, muzea a jiné kulturní služby*
- Příklad: D9BBA: *Pořízení vybavení pro ochranu sbírkových předmětů Arcibiskupského zámku v Kroměříži*
- Příklad: F3D77: *Revitalizace areálu klášterů Český Krumlov - oprava a rekonstrukce areálu bývalého kláštera sv. Kláry*
- Příklad: 6D3C6: *Obnova poutního areálu ve Staré Boleslavi – Mariánská zóna*

A.15 Finance

`finance-generic`: Společná kategorie pro finance

- CPV 66: *Finanční a pojišťovací služby*
- CPV 792: *Účetní, revizní a peněžní služby*
- CPV 794: *Podnikatelské a manažerské poradenství a související služby*

`finance-sluzby`: Finanční a pojišťovací služby

- CPV 66: *Finanční a pojišťovací služby*
- Příklad: 1B1D8: *Pojištění majetku a odpovědnosti za škodu MČ Praha 13*
- Příklad: 4FF8A: *Poskytnutí bankovního úvěru pro obec Vejprnice*

`finance-ucetni`: Účetní, revizní a peněžní služby

- CPV 792: *Účetní, revizní a peněžní služby*
- Příklad: C47B3: *Přezkum hospodaření Statutárního města Plzně za roky 2015 - 2017*
- Příklad: 05A06: *Operativní leasing vozidel*
- Příklad: 0C2E8: *Vedení účetnictví a mzdové agendy pro Léčebné lázně Lázně Kynžvart*

`finance-poradenstvi`: Podnikatelské a manažerské poradenství a související služby

- CPV 794: *Podnikatelské a manažerské poradenství a související služby*
- Příklad: 1F9C3: *Zajištění realizace projektu „Ústí nad Orlicí – Efektivní úřad“*
- Příklad: CACE6: *Procesní audit ÚP ČR a aktualizace typových pozic a vytvoření kompetenčních modelů*
- Příklad: 883BC: *Analýza a optimalizace podpůrných procesů VS ČR*

A.16 Právní a realitní služby

`legal-generic`: Obecná kategorie pro právní a realitní služby

- CPV 70: *Realitní služby*
- CPV 791: *Právní služby*
- CPV 7524: *Veřejná bezpečnost, právní a pořádkové služby*

`legal-reality`: Realitní služby

- CPV 70: *Realitní služby*
- Příklad: F3B18: *Zprostředkování prodeje zbytného majetku Ústeckého kraje*
- Příklad: FBF12: *Správa vybraných bytových a nebytových jednotek*

`legal-pravni`: Právní služby

- CPV 791: *Právní služby*
- Příklad: 890E8: *Právní služby: Kauza IZIP a.s.*
- Příklad: 4CAC6: *Právní služby a kontrola veřejných zakázek*

A.17 Technické služby

`techsluzby-generic`: Obecná kategorie pro technické služby

- CPV 51: *Instalační a montážní služby (mimo programového vybavení)*
- CPV 76: *Služby vztahující se k ropnému a plynárenskému průmyslu*
- CPV 90: *Kanalizace, odstraňování odpadu, čištění a ekologické služby*
- CPV 983: *Různé služby*
- CPV 45452: *Vnější úklidové práce*
- CPV 395258: *Úklidové hadry*

`techsluzby-odpady`: Kanalizace a odpad

- CPV 904: *Kanalizace*
- CPV 905: *Služby související s likvidací odpadů a odpady*
- Příklad: D88FB: *Sběr, svoz a likvidace odpadů*
- Příklad: E22C3: *Výběr provozovatele splaškové kanalizace a ČOV*

`techsluzby-cisteni`: Čistící, hygienické a enviromentální služby

- CPV 906: *Čistící a hygienické služby v městských a venkovských oblastech a související služby*

- CPV 909: *Čisticí a hygienické služby*
- CPV 907: *Environmentální služby*
- Příklad: EF3CF: *Zimní údržba silnic I.II.a III. třídy v Pardubickém kraji 2013-úsek 81/3-ZD*
- Příklad: 83EC4: *Informační systém kvality ovzduší v Kraji Vysočina*

`techsluzby-uklid`: Úklidové služby

- CPV 983: *Různé služby*
- CPV 45452: *Vnější úklidové práce*
- CPV 395258: *Úklidové hadry*
- Příklad: E6727: *Dodávka úklidových služeb*
- Příklad: 081DE: *Provoz a průběžný úklid veřejných toalet*

A.18 Výzkum

`vyzkum-generic`: Podpora výzkumu

- CPV 73: *Výzkum a vývoj a související služby*
- CPV 79315: *Společenskovědní výzkum*
- CPV 3897: *Výzkumné, testovací a vědecké technické simulátory*
- CPV 3829: *Výzkumné, hydrografické, oceánografické a hydrologické nástroje a přístroje*
- CPV 3012513: *Tonery pro střediska zpracování dat a výzkumná a dokumentační střediska*
- Příklad: 4E4DD: *Smluvní výzkum v oblasti katalýzy*
- Příklad: AD8CD: *Inovace výrobku a technologie výroby zakysaných smetan*

A.19 Marketing

`marketing-generic`: Reklamní a marketingové služby

- CPV 7934: *Reklamní a marketingové služby*
- CPV 79341: *Reklamní služby*
- CPV 793411: *Reklamní poradenství*
- CPV 793412: *Řízení reklamy*
- CPV 793414: *Reklamní kampaně*

- CPV 793415: *Letecká reklama*
- CPV 79342: *Marketingové služby*
- CPV 793421: *Přímý marketing*
- CPV 793422: *Propagační služby*
- CPV 793423: *Služby zákazníkům*
- CPV 7934231: *Průzkum zákazníků*
- CPV 79342311: *Průzkum spokojenosti zákazníků*
- CPV 7934232: *Péče o zákazníky*
- CPV 79342321: *Program věrných zákazníků*
- CPV 794: *Podnikatelské a manažerské poradenství a související služby*
- CPV 79413: *Poradenství v oblasti marketingového řízení*
- CPV 79416: *Práce s veřejností*
- CPV 794161: *Řízení práce s veřejností*
- Příklad: CFE60: *Zajištění reklamních ploch pro Národní divadlo*

A.20 Ostatní

`jine-pohostinstvi`: Pohostinství, ubytovací služby, pořádání konferencí

- CPV 55: *Pohostinství a ubytovací služby a maloobchodní služby*
- Příklad: 1DAA6: *Konferenční servis*

`jine-jidelny`: Služby závodních jídelen

- CPV 555: *Služby závodních jídelen a zabezpečování pohostinských služeb pro zákazníky*
- Příklad: 2B4E2: *Zajištění školního stravování ve školní jídelně v České Skalici*

`jine-admin`: Administrativní služby (správní, legislativní, bytové výstavby, cestovního ruchu atd.), pomocné kancelářské služby

- CPV 751: *Administrativní služby*
- CPV 795: *Pomocné kancelářské služby*
- Příklad: 96A56: *Programové a projektové řízení OPŽP*
- Příklad: 8360D: *Administrace specializačního vzdělávání zdravotnických pracovníků*

`jine-pruzkum`: Průzkum veřejného mínění a statistiky

- CPV 7931: *Průzkum trhu*
- CPV 7932: *Průzkum veřejného mínění*
- CPV 7933: *Statistické služby*
- CPV 793: *Průzkum trhu a ekonomický průzkum; průzkum veřejného mínění a statistiky*
- Příklad: ED948: *Domácí a příjezdový cestovní ruch – tracking 2019 - 2021*
- Příklad: FDC85: *Marketingový průzkum postojů populace a odborné veřejnosti k MZe a konzumaci médií*

`jine-opravy`: Opravy a údržba – obecná kategorie pro všechny druhy oprav stávajících zařízení

- CPV 50: *Opravy a údržba*
- Příklad: 98844: *Vozidla kategorie M1, N1 - oprava a servis*
- Příklad: D2E5A: *Oprava elektromotoru KSB*

B. Příloha: Analýza veřejných zakázek: tabulky

Tabulky v této kapitole popisují strukturu datové sady veřejných zakázek, s níž jsme pracovali v analýze a při trénování klasifikátoru. Uvádíme počet veřejných zakázek pro každou z kategorií, kterou máme k dispozici. Zároveň také zkoumáme průniky kategorií po dvou a počty dokumentů, které náleží do obou kategorií. Tyto informace jsou využity v sekci 3.1.1.

V tabulce B.1 jsou uvedeny počty dokumentů pro generické kategorie a jejich průniky po dvou. V tabulkách B.2 až B.16 máme uvedené počty dokumentů pro kategorie podle jednotlivých oblastí zájmu.

	celk	agro-generic	bezpecnost-generic	doprava-generic	energie-generic	finance-generic	it-generic	jidlo-generic	kancelar-generic	legal-generic	marketing-generic	prirodnizdroj-generic	remeslo-generic	social-generic	stav-generic	stroje-generic	techsluzby-generic	telco-generic	vyzkum-generic	zdrav-generic
agro-generic	8261	-	1	331	4	0	19	175	113	2	0	7	119	35	945	1376	115	1	2	4
bezpecnost-generic	1150	1	-	74	3	1	95	0	47	1	0	6	50	14	136	146	41	75	2	30
doprava-generic	10801	331	74	-	259	55	102	32	164	21	4	13	22	60	903	821	126	87	4	19
energie-generic	4079	4	3	259	-	6	10	4	20	6	3	8	5	7	625	307	111	9	2	19
finance-generic	2474	0	1	55	6	-	88	0	42	31	488	0	0	89	104	26	23	3	26	6
it-generic	15959	19	95	102	10	88	-	0	1375	18	113	4	26	139	606	925	219	1317	42	148
jidlo-generic	669	175	0	32	4	0	0	-	12	1	0	33	10	0	56	13	13	1	1	25
kancelar-generic	10651	113	47	164	20	42	1375	12	-	40	148	59	140	314	934	828	304	428	37	368
legal-generic	704	2	1	21	6	31	18	1	40	-	16	1	0	35	86	5	32	7	2	2
marketing-generic	1282	0	0	4	3	488	113	0	148	16	-	0	1	135	86	3	21	17	28	2
prirodnizdroj-generic	1190	7	6	13	8	0	4	33	59	1	0	-	30	0	87	64	23	1	4	134
remeslo-generic	1711	119	50	22	5	0	26	10	140	0	1	30	-	15	353	69	12	14	0	94
social-generic	3935	35	14	60	7	89	139	0	314	35	135	0	15	-	283	65	558	65	15	31
stav-generic	58528	945	136	903	625	104	606	56	934	86	86	87	353	283	-	2622	1683	486	67	283
stroje-generic	18508	1376	146	821	307	26	925	13	828	5	3	64	69	65	2622	-	546	643	293	646
techsluzby-generic	5352	115	41	126	111	23	219	13	304	32	21	23	12	558	1683	546	-	233	8	195
telco-generic	6115	1	75	87	9	3	1317	1	428	7	17	1	14	65	486	643	233	-	9	47
vyzkum-generic	1045	2	2	4	2	26	42	1	37	2	28	4	0	15	67	293	8	9	-	22
zdrav-generic	8931	4	30	19	19	6	148	25	368	2	2	134	94	31	283	646	195	47	22	-

Tabulka B.1: Počty zakázek pro kategorie oblasti `*-generic` a průniky kategorií po dvou

	celk	doprava-dily	doprava-generic	doprava-koleje	doprava-lidi	doprava-nakladni	doprava-opravy	doprava-osobni	doprava-pohotovost	doprava-posta	doprava-silnice	doprava-sluzby	doprava-special	doprava-udrzba
doprava-dily	328	-	328	0	0	13	19	2	0	0	2	0	1	9
doprava-generic	10801	328	-	289	346	3835	488	1457	881	69	1326	395	529	1002
doprava-koleje	289	0	289	-	2	9	4	0	0	0	0	0	0	2
doprava-lidi	346	0	346	2	-	5	5	11	0	0	0	0	6	0
doprava-nakladni	3835	13	3835	9	5	-	28	116	881	0	409	0	40	1002
doprava-opravy	488	19	488	4	5	28	-	11	15	0	1	0	1	1
doprava-osobni	1457	2	1457	0	11	116	11	-	12	0	6	0	529	5
doprava-pohotovost	881	0	881	0	0	881	15	12	-	0	0	0	8	0
doprava-posta	69	0	69	0	0	0	0	0	0	-	0	7	0	0
doprava-silnice	1326	2	1326	0	0	409	1	6	0	0	-	0	1	11
doprava-sluzby	395	0	395	0	0	0	0	0	0	7	0	-	0	0
doprava-special	529	1	529	0	6	40	1	529	8	0	1	0	-	3
doprava-udrzba	1002	9	1002	2	0	1002	1	5	0	0	11	0	3	-

Tabulka B.2: Počty zakázek pro kategorie oblasti `doprava-*` a průniky kategorií po dvou

	celk	it-generic	it-hw	it-opravy	it-servery	it-site	it-sw	it-sw-sluzby	it-system	it-vyvoj
it-generic	15959	-	6075	638	2881	377	5251	3982	2064	6572
it-hw	6075	6075	-	51	753	20	1076	247	94	357
it-opravy	638	638	51	-	67	7	77	75	39	91
it-servery	2881	2881	753	67	-	38	2881	466	296	748
it-site	377	377	20	7	38	-	48	103	47	377
it-sw	5251	5251	1076	77	2881	48	-	765	364	1138
it-sw-sluzby	3982	3982	247	75	466	103	765	-	772	3982
it-system	2064	2064	94	39	296	47	364	772	-	2064
it-vyvoj	6572	6572	357	91	748	377	1138	3982	2064	-

Tabulka B.3: Počty zakázek pro kategorie oblasti `it-*` a průniky kategorií po dvou

	celk	kancelar-bezpecnost	kancelar-cisteni	kancelar-generic	kancelar-nabor	kancelar-nabytek	kancelar-spotrebice	kancelar-stroje	kancelar-tisk
kancelar-bezpecnost	618	-	0	618	0	313	0	1	1
kancelar-cisteni	313	0	-	313	0	313	0	27	0
kancelar-generic	10651	618	313	-	342	4738	393	2150	1683
kancelar-nabor	342	0	0	342	-	0	0	0	1
kancelar-nabytek	4738	0	313	4738	0	-	393	112	133
kancelar-spotrebice	393	0	0	393	0	393	-	3	1
kancelar-stroje	2150	1	27	2150	0	112	3	-	48
kancelar-tisk	1683	1	0	1683	1	133	1	48	-

Tabulka B.4: Počty zakázek pro kategorie oblasti `kancelar-*` a průniky kategorií po dvou

	celk	stroje-elektricke	stroje-generic	stroje-laborator	stroje-prumysl
stroje-elektricke	2622	-	2622	276	234
stroje-generic	18508	2622	-	7639	7730
stroje-laborator	7639	276	7639	-	521
stroje-prumysl	7730	234	7730	521	-

Tabulka B.5: Počty zakázek pro kategorie oblasti `stroje-*` a průniky kategorií po dvou

	celk	telco-generic	telco-site	telco-sluzby	telco-tv
telco-generic	6115	-	1694	2568	1452
telco-site	1694	1694	-	41	61
telco-sluzby	2568	2568	41	-	0
telco-tv	1452	1452	61	0	-

Tabulka B.6: Počty zakázek pro kategorie oblasti `telco-*` a průniky kategorií po dvou

	celk	zdrav-generic	zdrav-leciva	zdrav-opravy	zdrav-osobni	zdrav-pristroje
zdrav-generic	8931	-	2014	500	450	6050
zdrav-leciva	2014	2014	-	4	23	54
zdrav-opravy	500	500	4	-	0	139
zdrav-osobni	450	450	23	0	-	32
zdrav-pristroje	6050	6050	54	139	32	-

Tabulka B.7: Počty zakázek pro kategorie oblasti `zdrav-*` a průniky kategorií po dvou

	celk	jidlo-generic	jidlo-potrava	jidlo-voda
jidlo-generic	669	-	394	42
jidlo-potrava	394	394	-	18
jidlo-voda	42	42	18	-

Tabulka B.8: Počty zakázek pro kategorie oblasti `jidlo-*` a průniky kategorií po dvou

	celk	prirodnizdroj-chemie	prirodnizdroj-generic	prirodnizdroj-pisky	prirodnizdroj-tezba
prirodnizdroj-chemie	804	-	804	0	2
prirodnizdroj-generic	1190	804	-	171	357
prirodnizdroj-pisky	171	0	171	-	171
prirodnizdroj-tezba	357	2	357	171	-

Tabulka B.9: Počty zakázek pro kategorie oblasti `prirodnizdroj-*` a průniky kategorií po dvou

	celk	energie-elektrina	energie-generic	energie-jina	energie-paliva	energie-sluzby
energie-elektrina	1459	-	1459	1459	125	1
energie-generic	4079	1459	-	3199	1567	283
energie-jina	3199	1459	3199	-	1567	19
energie-paliva	1567	125	1567	1567	-	4
energie-sluzby	283	1	283	19	4	-

Tabulka B.10: Počty zakázek pro kategorie oblasti `energie-*` a průniky kategorií po dvou

	celk	agro-generic	agro-les	agro-tezba	agro-zahrada
agro-generic	8261	-	4982	1029	2026
agro-les	4982	4982	-	1029	299
agro-tezba	1029	1029	1029	-	257
agro-zahrada	2026	2026	299	257	-

Tabulka B.11: Počty zakázek pro kategorie oblasti `agro-*` a průniky kategorií po dvou

	celk	remeslo-generic	remeslo-hudba	remeslo-odevy	remeslo-textil
remeslo-generic	1711	-	846	653	225
remeslo-hudba	846	846	-	1	2
remeslo-odevy	653	653	1	-	10
remeslo-textil	225	225	2	10	-

Tabulka B.12: Počty zakázek pro kategorie oblasti `remeslo-*` a průniky kategorií po dvou

	celk	social-generic	social-knihovny	social-kultura	social-pece	social-skoleni	social-vzdelavani	social-zdravotni
social-generic	3935	-	262	468	923	1168	117	129
social-knihovny	262	262	-	43	1	0	0	0
social-kultura	468	468	43	-	0	2	1	0
social-pece	923	923	1	0	-	8	2	10
social-skoleni	1168	1168	0	2	8	-	52	1
social-vzdelavani	117	117	0	1	2	52	-	1
social-zdravotni	129	129	0	0	10	1	1	-

Tabulka B.13: Počty zakázek pro kategorie oblasti `social-*` a průniky kategorií po dvou

	celk	finance-generic	finance-poradenstvi	finance-sluzby	finance-ucetni
finance-generic	2474	-	488	1684	336
finance-poradenstvi	488	488	-	8	16
finance-sluzby	1684	1684	8	-	10
finance-ucetni	336	336	16	10	-

Tabulka B.14: Počty zakázek pro kategorie oblasti `finance-*` a průniky kategorií po dvou

	celk	legal-generic	legal-pravni	legal-reality
legal-generic	704	-	498	207
legal-pravni	498	498	-	3
legal-reality	207	207	3	-

Tabulka B.15: Počty zakázek pro kategorie oblasti `legal-*` a průniky kategorií po dvou

	celk	techsluzby-cisteni	techsluzby-generic	techsluzby-odpady	techsluzby-uklid
techsluzby-cisteni	1949	-	1949	96	108
techsluzby-generic	5352	1949	-	1702	566
techsluzby-odpady	1702	96	1702	-	9
techsluzby-uklid	566	108	566	9	-

Tabulka B.16: Počty zakázek pro kategorie oblasti `techsluzby-*` a průniky kategorií po dvou

C. Příloha: Zdrojové kódy

Cílem této kapitoly je seznámit čtenáře se strukturou zdrojových kódů, které jsme použili jak pro trénování a testování modelů, tak pro predikci kategorií na produkčním webu. Nejprve popíšeme adresářovou strukturu celého řešení. V další sekci projdeme jednotlivé moduly podle logické návaznosti úkolů. Od čtení textů smluv a zakázek, předzpracování textu a trénování modelů se přesuneme až k finálnímu produktu, serveru s REST interface, který používají ostatní části produkčního webu Hlídače státu pro zpracování smluv. Předposlední část obsahuje popis skriptů, které je možno použít pro zprovoznění běhového prostředí projektu, buď pomocí *virtualenv*, nebo Dockeru. Přesné instrukce pro spuštění REST serveru pomocí Dockeru se nachází v závěru přílohy.

C.1 Výběr programovacího jazyka

Při výběru vhodného řešení klasifikace, jemuž se věnuje kapitola 4, jsme museli řešit i výběr konkrétních implementací. Tomu předcházelo rozhodování, který programovací jazyk pro implementaci projektu použijeme; cílem bylo pro všechny kroky řešení použít jednotné prostředí. Je pravda, že ve fázi testování modelů by toto nebylo vyloženě nutné, a bylo by možné jednotlivé klasifikátory spouštět za pomocí jednoduchých skriptů napsaných například v Unix shellu. Pro finální nasazení na serverech Hlídače státu by však toto řešení nebylo dostačující.

Protože jsme během práci na projektu chtěli experimentovat s různými možnostmi, důležitým kritériem při výběru jazyka byla rychlost vývoje. Efektivita a rychlost běhu byly méně důležité než rychlost vývoje, protože při testování i při běhu na produkci programy běží dávkově. Posledním kritériem při rozhodování byla dostupnost knihoven pro práci s textem a strojové učení.

První rozhodovací podmínka vyřadila „nízkoúrovňové“ jazyky jako C, C++ a další. Ve výběru zůstaly *managed* jazyky jako C# nebo Java, případně interpretované jazyky jako Python nebo Ruby. Při posuzování jazyků podle zbývajících dvou kritérií se ukázaly přednosti jazyka Python: má k dispozici velké množství potřebných knihoven, ty jsou však typicky implementované v C nebo C++, takže dostatečná efektivita je zachována.

S výjimkou některých spouštěcích skriptů jsou proto zdrojové kódy projektu implementovány v jazyce Python. Jako interpreter jsme použili CPython, verzi 3.7. Alternativní implementace Pythonu (například PyPy) jsme nepoužili, protože mnoho navázaných knihoven je podporuje buď experimentálně, nebo vůbec.

C.2 Adresářová struktura

Zde uvádíme základní popis jednotlivých adresářů v projektu.

- `pylib`: obsahuje zdrojové kódy v jazyce Python.

  - `pylib/dp2`: veškeré zdrojové kódy modulů jsou uloženy v hierarchii podadresáři `dp2`, aby tak byly součástí samostatného modulu `dp2`.[1]
  - `pylib/dp2/exec`: spustitelné pythonové skripty jsou uloženy zde.
  - `pylib/dp2/rest`: vstupní skripty pro servery s REST interface.
  - `pylib/dp2/exec_utils`: obsahuje pomocné soubory pro spustitelné pythonové skripty.
- `big`: adresář je určen pro velké soubory, které nebyly určeny pro uložení do verzovacího systému.
  - `big/models`: výchozí cesta pro uložení natrénovaných modelů.
- `models`: cesta pro uložení natrénovaných modelů, které jsme přidali do verzovacího systému.
- `cpv`: zdrojové soubory pro převod CPV kódů veřejných zakázek na kategorie.
- `api`: zdrojové soubory popisující API webového serveru.
- `run`: obsahuje soubory a skripty pro spouštění zdrojových kódů a jejich nahrání na produkční servery.
  - `run/venv`: skripty pro vytvoření pythonovského virtuálního prostředí (*virtualenv*).
  - `run/docker`: skripty pro vytvoření dockerového obrazu a jeho nahrání na produkční servery.
  - `run/failed_inputs`: speciální adresář, kam se ukládají chybné vstupy z REST interface.

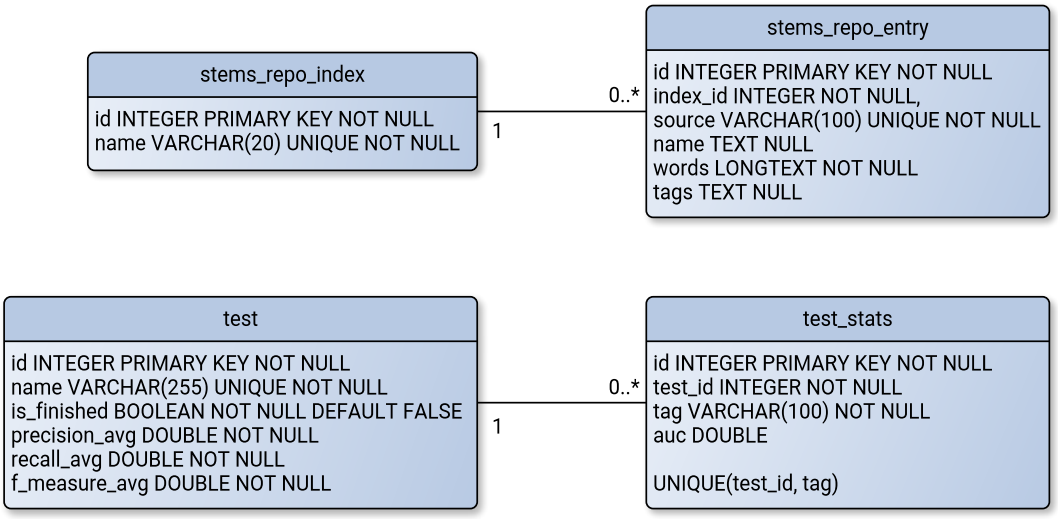
C.3 Externí systémy

Funkčnost některých částí zdrojového kódu závisí na dvou externích systémech: v první řadě jde o server Apache Elasticsearch, na němž byly původně uložené smlouvy a veřejné zakázky, v druhé řadě jde o relační databázi MySQL, kam jsme ukládali zpracované texty a výsledky testování. Webový server nicméně funguje samostatně a žádný z těchto systémů nevyžaduje.

Elasticsearch umožňuje ukládat dokumenty ve formátu JSON. Každý dokument musí být zařazený do *indexu*; jeden index typicky obsahuje dokumenty stejného druhu. Instance Elasticsearch, kterou máme k dispozici, má dva indexy, obsahující všechny smlouvy a veřejné zakázky, včetně textů příloh.

UML databázové schéma je znázorněno na obázku C.1. Pro ukládání zpracovaných textů se v databázi nachází dvě tabulky. Tabulka `stems_repo_index` obsahuje seznam indexů, do nichž ukládáme texty. Význam slova *index* je v tomto případě identický s významem u Elasticsearch; tedy i v tomto případě máme dva indexy. Samotné zpracované texty se ukládají do tabulky `stems_repo_entry`.

[1] Zkratka *dp2* znamená *diplomová práce, druhá verze* – tato verze vznikla po zásadním refactoringu.



Obrázek C.1: UML diagram databázových tabulek

Každý řádek v této tabulce odpovídá jednomu dokumentu. Ukládáme unikátní identifikátor dokumentu (`source`), identifikátor indexu, slova převedená na základní tvar, která se v dokumentu nachází, a seznam kategorií, pokud je pro dokument k dispozici. Pro přehlednost při vyhledávání je do tabulky možné uložit jméno dokumentu (`name`).

Pro ukládání výsledků testů se v databázi používají další dvě tabulky. Každý řádek v tabulce `test` obsahuje výsledky jednoho testu, například průměrnou precision, recall atd. Výsledky pro metriku AUC se však ukládají v samostatné tabulce `test_stats` – z toho důvodu, že metrika AUC je definována samostatně pro každou kategorii.

C.4 Popis modulů a tříd podle funkcí

V následujím textu představíme jednotlivé části zdrojového kódu podle logických kroky v projektu, od získávání textů smluv a zakázek po výrobu webového serveru.

C.4.1 Shromáždění textu a předzpracování

Předpokládáme, že máme k dispozici server Elasticsearch; v něm jsou v indexu `hlidacsmluv` uloženy smlouvy a v indexu `zakazky` uloženy zakázky ve formátu JSON. Dále je třeba mít k dispozici databázový server MySQL. Třídy popsané v této části odpovídají UML diagramu na obrázku C.2.

Pro přístup k indexům používáme třídy `contracts_repo.ElasticContracts` a `tenders_repo.ElasticTenders`. Účelem těchto tříd je umožnit iteraci nad uloženými dokumenty. Iteraci obou druhů dokumentů provádíme v náhodném pořadí; díky tomu později, při rozdělování dokumentů na trénovací a testovací sadu, již stačí vybrat dokumenty v sériích.

Každý získaný dokument ve formátu JSON je převeden na datovou strukturu `contract.Contract`, resp. `tender.Tender`. Tyto struktury jsou určeny pro

reprezentaci extrahovaných dat a obsahují metodu `all_text`, která vrátí jediný řetězec s textovým obsahem dokumentu.

Dalším úkolem je získaný textový řetězec předzpracovat a extrahovat slova (viz sekce 4.3.2). Tyto úkoly provádí třída `stem.UdPipeStemmer`. Třída však netvoří přímo výsledné n-gramy; namísto toho vrací pole s jednotlivými tokenizovanými slovy, oddělenými hodnotou `None` (logický ekvivalent `NULL` v jazyce C), pokud se mezi tokenizovanými slovy vyskytlo vyřazené slovo. Tento postup umožňuje v případě potřeby vytvořit n-gramy o jiném n než 3, aniž by bylo třeba provádět extrakci znovu.

Extrahovaná slova se ukládají do databázové tabulky `stems_repo_entry`. Třídy `StemsRepo` a `StemsIndex` poskytují abstrakci nad úložištěm zpracovaných dokumentů. Do tabulky se také (v případě zakázek) ukládají jména kategorií, do nichž dokument náleží. Mapování mezi CPV kódy a kategoriemi je k dispozici v modulu `tag_cpv_info`.

Pro zrychlení zpracování dat v tomto kroku jsme celý proces paralelizovali: skript `stem_all_contracts_dispatcher` se spustí právě jednou a postupně vybírá všechny smlouvy z Elasticsearch. Skript `stem_all_worker` může být spuštěn vícekrát, pomocí REST interface přijímá z `dispatcher` smlouvy, provede předzpracování textu a uloží výsledek do databáze. Tímto způsobem jsme schopni zajistit, že každá smlouva se zpracuje právě jednou. (Pro veřejné zakázky analogicky používáme skript `stem_all_tenders_dispatcher`).

C.4.2 Manuální označení smluv

Před prováděním experimentů (kapitola 5) jsme ručně prošli a označili správnými kategoriemi 204 smluv. Toto manuální označení je implementováno v modulu `manually_tagged` a spouští se pomocí skriptu `exec.manually_tag`.

C.4.3 Reprezentace modelů

UML diagram tříd uvedených v následujících dvou sekcích je na obrázku C.3.

Pro trénování modelů jsme použili několik externích knihoven: `facebookresearch/fastText` jako implementaci modelu FastText[2], `gensim` pro implementaci modelu Doc2Vec[3] a `scikit-learn` pro implementaci random forests[4]. Pouze metoda klíčových slov je implementována přímo v našem kódu. Vzhledem k různému původu modelů bylo třeba nad nimi vytvořit abstrakční vrstvu. K tomuto účelu slouží třídy `model.Model`, `model.VectorModel` a `model.StemModel`. Tyto třídy pouze specifikují metody, které musí obsahovat jejich podtřídy; chovají se tedy jako interfaces v C# nebo Javě.

Třída `Model` obsahuje dvě důležité metody: `save(path)` pro uložení natrénovaného modelu na disk a `load(path)` pro načtení modelu z disku. Argument `path` označuje adresář, z něhož je model načítán, případně kam je ukládán.

Třída `StemModel` je podtřídou `Model` a implementují ji modely, které přímo predikují kategorie dokumentu a jejich pravděpodobnosti.

[2] `https://github.com/facebookresearch/fastText`

[3] `https://radimrehurek.com/gensim/`

[4] `https://scikit-learn.org/`

Metoda `train(transformer, stem_entries)` spouští trénink metody. Argument `stem_entries` musí být iterátor, který vrací instance třídy `types.StemEntry`, obsahující pole s tokenizovanými slovy a seznam kategorií (viz sekce C.4.1); `transformer` obsahuje implementaci třídy `stem.StemTransformer`, která umožňuje převod polí s tokenizovanými slovy na n-gramy. Metoda `predict(transformer, stem)` predikuje kategorie pro předané pole s tokenizovanými slovy a vrátí tento výsledek jako slovník, kde klíči jsou jména kategorií a hodnotami jsou pravděpodobnosti, že dokument náleží do příslušné kategorie.

Třída `VectorModel` je rovněž podtřídou `Model`. Jediným rozdílem oproti třídě `StemModel` je návratová hodnota metody `predict` – je vrácen číselný vektor reprezentující předaný dokument. Tato třída se používá v podpůrných třídách pro metodu random forests, která potřebuje dokumenty nejprve převést na vektory.

Veškeré konkrétní implementace modelů jsou dostupné v modulu `model`.

C.4.4 Trénování modelů

Během trénování jsme modely ukládali do adresáře `big/models`. Jako správce uložených modelů slouží třída `models_repo.GitModelsRepo`, implementace abstraktní třídy `models_repo.ModelsRepo`. Třída specifikuje kořenový adresář pro uložené modely. Dále při ukládání (metoda `save(model, model_id)`) zajišťuje, že v adresáři s uloženým modelem se vytvoří soubory `_module` a `_class`, označující modul a třídu, která natrénovaný model ukládá. Díky tomu je možné při volání metody `load(model_id)` načíst správnou třídu modelu bez její explicitní specifikace.

Instrukce, jak modely trénovat, na jakých datech a parametrech, jsou umístěné v modulu `models_cookbook`. Ve třídě `MyModelsCookbook` tvoříme pole *receptů* – každý recept je funkce, která načte příslušnou datovou sadu, natrénuje model a uloží jej na disk.

Vzhledem k problémům, které má CPython s vícevláknovým zpracováním (viz sekce 6.4.4), spouštíme trénování každého modelu v samostatném procesu. Trénovací procesy postupně spouští skript `exec.cook_models`.

C.4.5 Testování modelů

Třídy pro testování modelů jsou uvedené v UML diagramu na obrázku C.4.

Při testování natrénovaného modelu je třeba jeho výsledek dodatečně upravit. Vzhledem k tomu, že se za predikci považuje pět kategorií s nejvyšší hodnotou pravděpodobnosti, je třeba ve výsledku ponechat jen těchto pět kategorií. K těmto úpravám slouží `combiners.ModelCombiner`, tato třída nechá výsledek predikce modelu projít řadou funkcí, které upravují výsledek. Konkrétní funkce specifikuje uživatel třídy; námi používané funkce se nachází ve třídě `combiners.CombiningFunctions`.

Samotné nástroje pro testování se nechází v modulu `test`. Jádrem testovacího procesu je třída `test.StemModelTest`, která v konstruktoru přijímá příslušný `ModelCombiner`, `StemTransformer` a iterátor testovacích dokumentů. Metoda `iter_real_predicted_results()` vrací iterátor, který pro každý testovací dokument obsahuje seznam skutečných kategorií a seznam predikovaných kategorií.

Tento iterátor přijímají třídy `metrics10`, `auc_per_tag` a další; úkolem těchto tříd je vypočítat metriky úspěšnosti jednotlivých modelů.

Jako abstrakce pro ukládání výsledků testů slouží dvojice abstraktních tříd `test.Test` a `test.TestsRepo`. Třída `TestsRepo` představuje úložiště pro ukládání výsledků. Její metoda `load(test_name)` vrací instanci třídy `Test`, která shromáždí výsledky testování na základě přijatého iterátoru a uloží je. V našem případě ukládáme výsledky testů do databáze; konkrétními implementacemi jsou třídy `test.DbTest` a `test.DbTestsRpo`.

I v této fázi máme instrukce k testování modelů umístěné v samostatném modulu `test_cookbook`. Každý test běží v samostatném procesu (skript `exec.cook_tests`).

C.4.6 REST server

Diskuze k řešení REST serveru se nachází v sekci 6.4.4. Připomeňme, že pro implementaci jsme vybrali framework Flask. Základní částí serveru je modul `rest.server_v6`. Pro zajištění funkčnosti si modul nahrává velké množství tříd, uvedených v předešlých sekcích. Mimo jiné se jedná o `UdPipeStemmer` (tokenizace textu), podtřídu `StemModel` (vybraný model) a `ModelCombiner` (dodatečná úprava predikce modelu). Na obrázku C.5 znázorňuje UML graf navázané třídy.

Celý proces klasifikace smlouvy je rozdělen do tří částí. Každé části odpovídá jeden REST endpoint:

- `GET /stemmer`: přijme na vstupu smlouvu ve formátu JSON a vrátí pole s tokenizovanými slovy.
- `GET /classifier`: přijme pole s tokenizovanými slovy a vrátí slovník, kde pro každou kategorii je uvedena hodnota predikce. Tyto hodnoty však zatím nejsou zpracované pomocí `ModelCombiner`.
- `GET /finalizer`: přijme slovník s predikovanými hodnotami, zpracuje je pomocí `ModelCombiner` a vrátí konečný výsledek klasifikace.

Důvodem pro rozdělení celého procesu do tří endpointů je umožnění kešování tokenizovaných slov (tokenizace je časově nejnáročnější část procesu) a snazší možnost ladění chyb.

API serveru je specifikováno pomocí formátu OpenAPI. Samotná specifikace ve formátu YAML je přístupná v repozitáři zdrojových kódů v adresáři `api`. Ve spuštěném serveru je však také k dispozici webový interface na adrese `/api`. Přes tento interface je možné prohlížet specifikace jednotlivých endpointů a spouštět testovací volání. Ukázka interface je na obrázku 6.1.

Pro ladění modelů je k dispozici také HTML endpoint `/explain`. Formulář dokáže načíst smlouvu ve formátu JSON a zobrazit nalezená klíčová slova.

C.5 Běhové prostředí a nasazení kódu

V sekci 6.3 jsme se věnovali možnostem nasazení zdrojového kódu. Rozhodli jsme se podporovat dva možné scénáře: *virtualenv* pro vývoj a ladění kódu a

Docker pro nasazení projektu na produkčním prostředí. Podpora je realizována shellovými skripty v adresáři `run`.

Pro oba scénáře je možné konfigurovat spouštěné programy pomocí souboru `run/env`, obsahující proměnné prostředí.

Pro vytvoření *virtualenv* je třeba vytvořit virtuální prostředí a nainstalovat knihovny, na kterých zdrojový kód projektu závisí. Oba tyto kroky je možné provést pomocí skriptu `venv/prepare_venv.sh`.

Při použití Dockeru je nutné nejprve lokálně sestavit obraz projektu, což je možné provést skriptem `docker/build.sh`. Tento skript spouští nástroj `docker build` s příslušnými parametry; sestavování obrazu je řízené souborem `docker/Dockerfile`. Skript `docker/launch_server_v6.sh` následně slouží k lokálnímu spuštění obrazu.

Pro spuštění sestaveného obrazu na produkčních serverech je třeba být přihlášený na Docker účet `hlidacstatu` a mít SSH přístup k produkčním serverům; zároveň se kontroluje, zda veškeré změny v kódu byly nahrány do verzovacího systému. Nejprve je třeba nahrát obraz do repozitáře Docker Hub, k čemuž slouží skript `docker/push.sh`. Samotné spuštění na produkčních serverech se provádí skriptem `docker/deploy_hlidac.sh`. Tento skript se připojí na každý produkční server, stáhne z Docker Hubu novou verzi obrazu, spustí ji a otestuje, že REST interface je k dispozici.

C.6 Instrukce ke spuštění REST serveru

V této sekci uvádíme přesný postup pro zprovoznění REST serveru. Tento návod předpokládá:

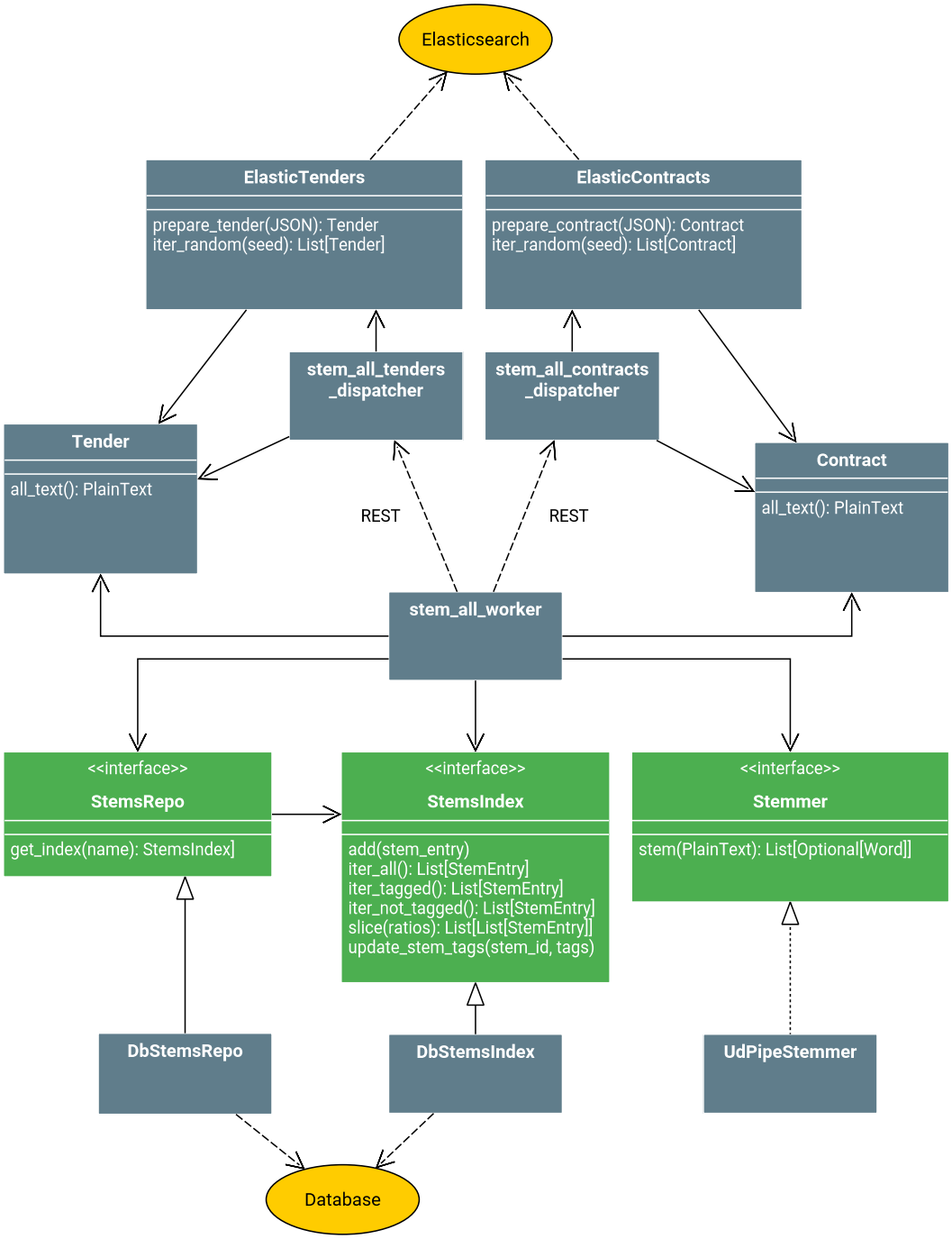
- na stroji, kde chceme server spustit, běží systém unixového typu,
- na stroji je nainstalován Docker (používáme verzi 19.03),
- repozitář zdrojových kódů, dostupný v příloze práce, je k dispozici.

Nyní je třeba provést následující kroky:

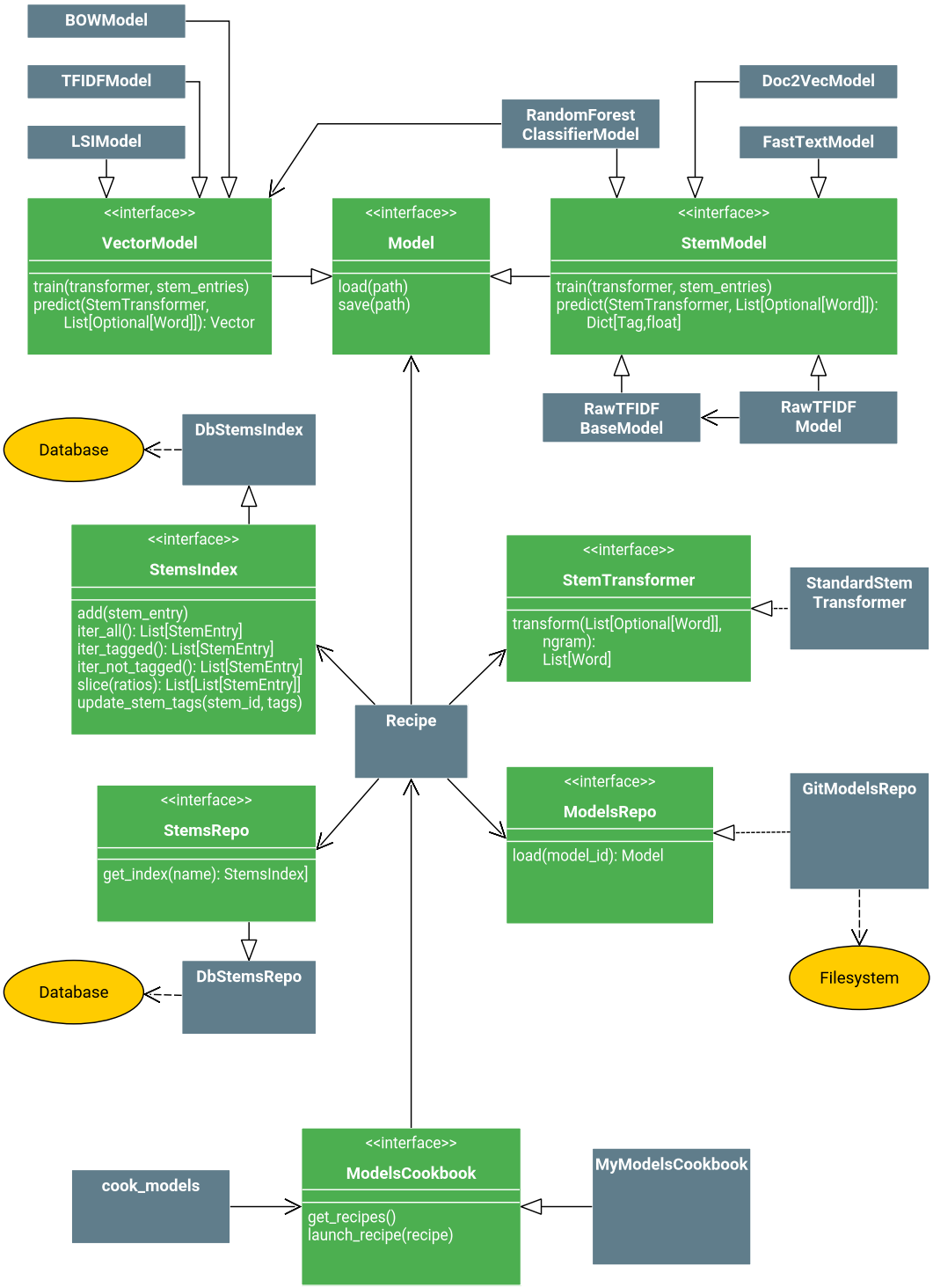
1. V souboru `run/docker/env` volitelně upravíme proměnnou `DOCKER_SERVER_PORT` – po spuštění obrazu bude na tomto portu k dispozici REST server.
2. Spustíme skript `run/docker/build.sh`. Tento skript použije soubor `run/docker/Dockerfile` ke kompilaci a sestavení obrazu `hlidacstatu/klasifikace-smluv:v6dp`.
3. Spustíme skript `run/docker/launch_server_v6.sh`. Tento skript spustí sestavený obraz.
4. Na výstupu skriptu se po spuštění kontejneru spustí jeho logovací výstup.
5. Na portu `localhost:$DOCKER_SERVER_PORT` je nyní přístupný REST server.
6. Logovací výstup je možné zastavit pomocí zkratky `Ctrl-C`, ovšem server stále poběží na pozadí. Pro zastavení je třeba spustit `run/docker/stop.sh`.

Se spuštěným serverem je možné například:

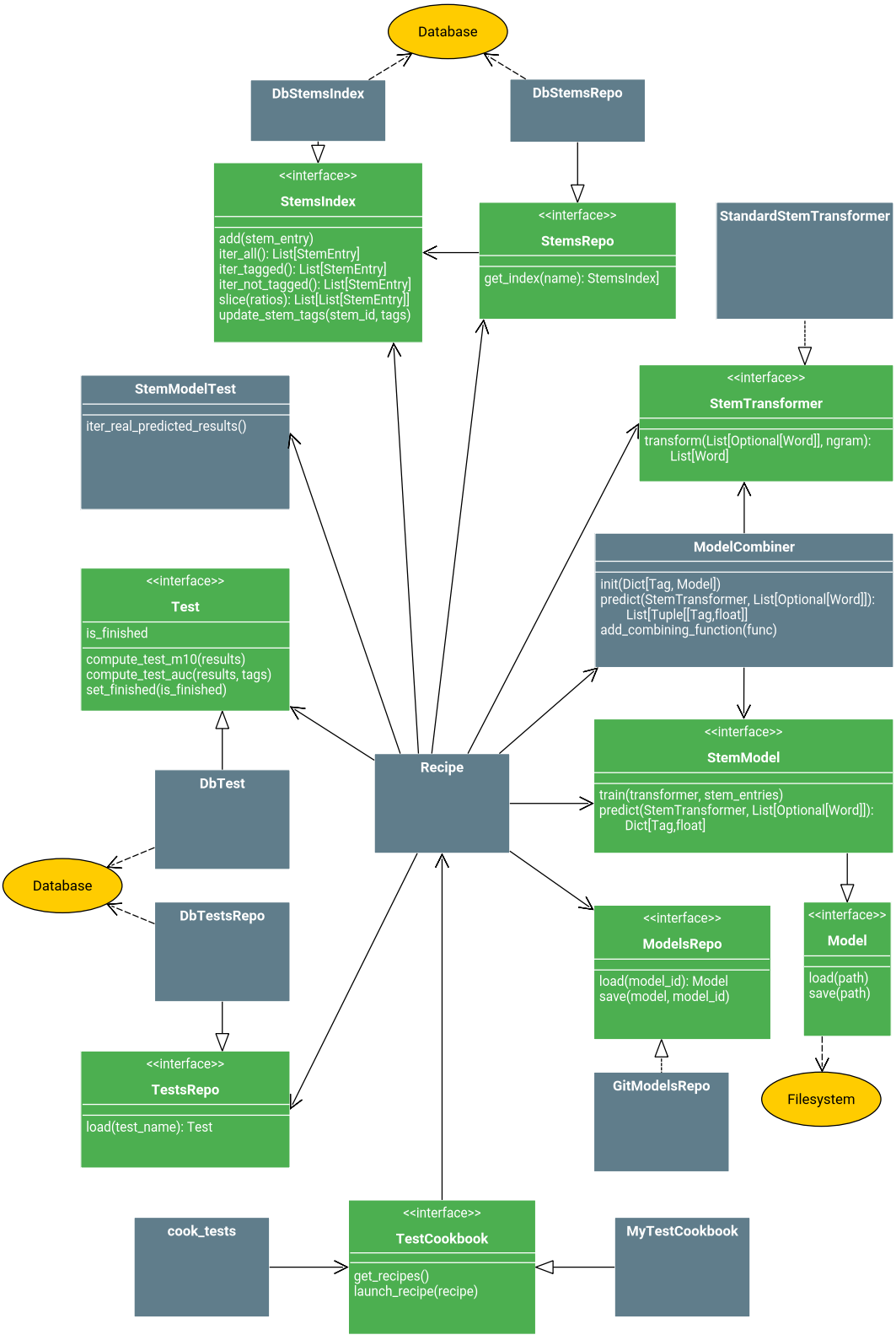
- testovat REST API volání (na `localhost:$DOCKER_SERVER_PORT/api`),
- vkládat JSON smluv do HTML endpointu `localhost:$DOCKER_SERVER_PORT/explain` a zobrazit si predikci klasifikátoru – výběr smluv ve formátu JSON je rovněž dostupný v elektronické příloze.



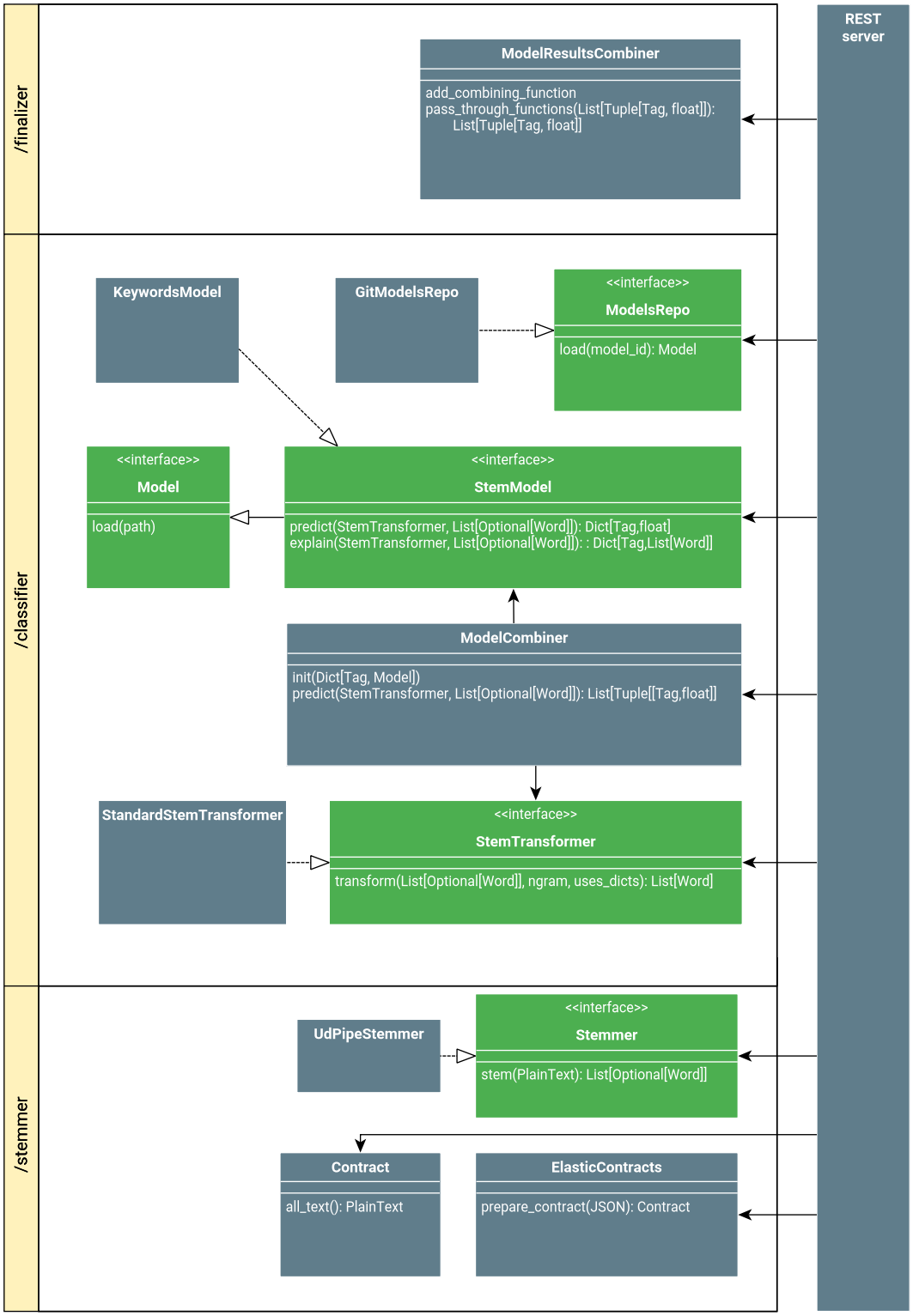
Obrázek C.2: UML class diagram tříd, používaných při předzpracování textu



Obrázek C.3: UML class diagram tříd, používaných při trénování modelů



Obrázek C.4: UML class diagram tříd, používaných při testování modelů



Obrázek C.5: UML class diagram tříd, s nimiž pracuje REST server