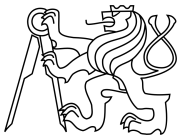
ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ



ZADÁNÍ DIPLOMOVÉ PRÁCE

Název:	Učení domén pojmenovaných entit
Student:	Bc. Tomáš Benák
Vedoucí:	Ing. Milan Dojčinovski
Studijní program:	Informatika
Studijní obor:	Webové a softwarové inženýrství
Katedra:	Katedra softwarového inženýrství
Platnost zadání:	Do konce letního semestru 2016/17

Pokyny pro vypracování

V textových dokumentech se často vyskytuje velký počet pojmenovaných entit. Nicméně ne všechny tyto entity jsou předmětem zájmu uživatele. Uživatel může mít zájem jen o entity z určitého typu domény, jako je například doména sport, hudba nebo politika. Cílem této práce je analyzovat a popsat možnosti učení, navrhnout a implementovat model pro učení domén.
Pokyny:
- Seznamte se s aktuálním stavem a proveďte rešerši.
- Analyzujte existující datasety a možnosti pro učení domén entit a případně vytvořte vlastní dataset na základě Wikipedia a DBpedia.
- Navrhněte a implementujte několik modelů pro učení domén entit.
- Navrhněte a implementujte rozhraní pro automatické učení domén entit v podobě služeb REST.
- Proveďte experimenty na různých modelech a pomocí různých učících algoritmů.

Seznam odborné literatury

Dodá vedoucí práce.

L.S.

Ing. Michal Valenta, Ph.D.
vedoucí katedry

prof. Ing. Pavel Tvrdík, CSc.
děkan

V Praze dne 11. prosince 2015

České vysoké učení technické v Praze
Fakulta informačních technologií
Katedra softwarového inženýrství

Diplomová práce

Učení domén pojmenovaných entit

Bc. Tomáš Benák

Vedoucí práce: Ing. Milan Dojčinovski

8. května 2016

Poděkování

Touto formou bych rád poděkoval Ing. Milanu Dojčinovskému za odborné vedení mé diplomové práce a pomoc s orientací v dané problematice. Dále děkuji své rodině a přítelkyni za podporu po celou dobu mého studia.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval(a) samostatně a že jsem uvedl(a) veškeré použité informační zdroje v souladu s Metodickým pokynem o etické přípravě vysokoškolských závěrečných prací.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona, ve znění pozdějších předpisů. V souladu s ust. § 46 odst. 6 tohoto zákona tímto uděluji nevýhradní oprávnění (licenci) k užití této mojí práce, a to včetně všech počítačových programů, jež jsou její součástí či přílohou, a veškeré jejich dokumentace (dále souhrnně jen „Dílo“), a to všem osobám, které si přejí Dílo užít. Tyto osoby jsou oprávněny Dílo užít jakýmkoli způsobem, který nesnižuje hodnotu Díla, a za jakýmkoli účelem (včetně užití k výdělečným účelům). Toto oprávnění je časově, teritoriálně i množstevně neomezené. Každá osoba, která využije výše uvedenou licenci, se však zavazuje udělit ke každému dílu, které vznikne (byť jen zčásti) na základě Díla, úpravou Díla, spojením Díla s jiným dílem, zařazením Díla do díla souborného či zpracováním Díla (včetně překladu), licenci alespoň ve výše uvedeném rozsahu a zároveň zpřístupnit zdrojový kód takového díla alespoň srovnatelným způsobem a ve srovnatelném rozsahu, jako je zpřístupněn zdrojový kód Díla.

V Praze dne 8. května 2016 .....

České vysoké učení technické v Praze
Fakulta informačních technologií

Odkaz na tuto práci

Benák, Tomáš. *Učení domén pojmenovaných entit.* Diplomová práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2016.

Abstrakt

Diplomová práce se zabývá doménami pojmenovaných entit a možnostmi strojového učení nad nimi. Práce nejprve analyzuje problém strojového učení, zdrojů dat a dosavadních řešení. Na základě těchto analýz je navrhnuta a implementována aplikace sloužící k tvorbě trénovacích datasetů a REST služba automatizující proces učení domén entit. Dále je představen nástroj Weka, který vypomáhá s vytvořením natrénovaných modelů, a projekt DBpedia, který je hlavním zdrojem pojmenovaných entit. Nakonec jsou provedeny experimenty k vyhodnocení kvality vytvořených modelů pro učení domén pojmenovaných entit.

Klíčová slova Strojové učení, Domény, Pojmenované entity, REST API, Weka, DBpedia

Abstract

The master thesis deals with the domains of the named entities and the possibilities of machine learning over them. At first the thesis analyses the problem of machine learning, the sources of data and the actual solutions. Based on these analyzes, the application, which creates the training datasets, and the REST API, which automates the process of learning domains for entities,

are designed and implemented. Furthermore, the program Weka, which helps with creating models, and the project DBpedia, which is the main source of named entities, are described. Finally, the experiments are made to evaluate the quality of created models for learning domains for named entities.

Keywords Machine Learning, Domains, Named Entities, REST API, Weka, DBpedia

Obsah

Seznam obrázků

Seznam tabulek

Úvod

Motivace

Každý uživatel procházející internetové stránky přijde do styku s velkým množstvím strukturovaných i nestrukturovaných informací, které obsahují pojmenované entity. Na základě těchto entit se dále rozhoduje, zda je pro něj text zajímavý nebo ne. Příkladem mohou být entity *volby* nebo *lední hokej*, na které může uživatel narazit při procházení stránek zpravodajských serverů, které obsahují články například o prezidentských volbách nebo blížícím se mistrovství světa v ledním hokeji. S rozvojem strojového učení, které se v této době spíše zaměřuje na rozšířenou realitu nebo umělou inteligenci, je možné pojmenované entity zpracovávat a provádět nad nimi různá učení. Důležitou vlastností těchto pojmenovaných entit je zařazení do nějaké oblasti zájmu, neboli domény. V konkrétním případě našich entit *volby* a *lední hokej* se jedná o domény *politika*, respektive *sport*. A právě tyto domény jsou pro uživatele zajímavé, protože mu mohou rozhodnutí o zajímavosti textu usnadnit.

Díky výskytu velkého množství pojmenovaných entit v textových dokumentech byl proces jednoduché identifikace jejich domény hlavní motivací této práce. Tento proces by měl přispět k lepšímu porozumění textovým dokumentům. Dalším podnětem bylo navázání na projekt *DBpedia Domains: augmenting DBpedia with domain information* [1], který se zabývá obdobným problémem, ovšem za použití jiných metod, které se ukázaly jako nepříliš úspěšné.

Cíle práce

Hlavním cílem této diplomové práce je navrhnout a implementovat nástroje, které budou schopny vygenerovat modely pro učení domén entit, ale také provést automatické učení nad těmito modely v podobě REST služeb. Tento návrh bude založen na analýze existujících datasetů pojmenovaných entit, ana-

lýze možností strojového učení a také na analýze dosavadních řešení. Součástí hlavního cíle této práce je provést experimenty nad vygenerovanými modely, za účelem vyhodnocení jejich kvality a použitelnosti.

Struktura práce

Práce je rozdělena do čtyř kapitol. V první kapitole *Rešerše a vymezení pojmů* jsou popsány základní pojmy spojené s touto prací a provedena rešerše dosavadních řešení. Druhá kapitola, která má název *Analýza*, popisuje problém učení domén pojmenovaných entit s analýzou zdrojů domén. Dále analyzuje existující datasety pojmenovaných entit a algoritmy strojového učení. A v poslední řadě popisuje požadavky práce a rozebírá pojem REST služby. Ve třetí kapitole s názvem *Návrh a implementace* jsou představeny použité technologie a popsán postup návrhu a implementace aplikace na tvorbu trénovacích datasetů a REST API. Poslední čtvrtá kapitola *Experimenty* popisuje cíle, konfigurace a vyhodnocení experimentů, které byly provedeny nad modely určenými k učení domén entit z druhé a třetí kapitoly této práce.

Kapitola 1

Rešerše a vymezení pojmů

Ke správnému pochopení problému a pojmů, které souvisí s touto prací, budou v této kapitole představena data, se kterými se bude pracovat, a také vysvětlen termín strojové učení, na němž je tato práce postavena. Z hlediska motivace a použitelnosti této práce zde bude představeno dosavadní řešení a řešení zpracovávající podobné odvětví tohoto problému.

1.1 Data

Práce se zabývá hlavně textovými dokumenty a entitami v nich. Tato část se zaměřuje na zdroj, formát a reprezentaci dat, které budou využity pro účely této práce.

1.1.1 Sémantický web

Sémantický web z obecného pohledu představuje rozšíření webu stávajícího. Hlavní myšlenkou je vložit do webových stránek přidanou hodnotu ve formě významu, o kterém informace pojednávají. Tento problém už v roce 2001, popsal Tim Berners-Lee [2] a upozorňuje na fakt, že web se stále rozšiřuje o webové stránky, ve kterých je těžší nalézt relevantní informace. Základním kamenem sémantického webu jsou technologie Resource Description Framework (RDF), Web Ontology Language (OWL) a Extensible Markup Language (XML), které umožňují popsat věci a přidat jim tak jejich daný význam. Popisy věcí přitom mohou být součástí HTML[1] dokumentů, a tak zajistit zachování uložení dokumentu jako jednoho velkého celku. Z druhého pohledu lze obě části těchto typů informací od sebe oddělit, jako je to například u Wikipedie a DBpedie z podkapitoly 2.4. Tato práce se bude zabývat hlavně přístupem k takto strukturovaným datům s tím, že využije právě zmíněný formát RDF.

[1] HTML – HyperText Markup Language – Značkovací jazyk používaný jako standard při vytváření webových stránek.

Resource Description Framework (RDF) patří do specifikace World Wide Web Consortium (W3C)[2]. Jeho datový model patří do rodiny konceptuálních modelů a jeho hlavní myšlenkou je práce s informacemi o zdrojích. Zdroji jsou myšleny všechny elementy, mezi něž můžeme zařadit i pojmenované entity, které je možné popsat unikátním identifikátorem (URI[3]). Každý zdroj s identifikátorem RDF popisuje jednoduchými výrazy ve formě trojic subjekt – predikát – objekt. Každou složkou této trojice je buď jiný zdroj, slovníkový term, anebo přímo hodnota, kterou vyjadřuje. Takto vytvořenou trojici lze vyjádřit v přirozeném jazyce srozumitelnou větou s podmětem, přísudkem a předmětem. Například z jednoduché slabikářové věty „Máma mele maso.“ převedené do formátu RDF dostaneme trojici, kde subjektem je „Máma“, predikát představuje slovo „mele“ a objekt tvoří slovo „maso“.

Strukturovaná data nabízí lidem nejen dotazování, obdobně třeba jako nad relačními databázemi pomocí jazyka SPARQL, ale i vytváření sofistikovaných algoritmů, které umožní například umělé inteligenci porozumět významu informací anebo jako v případě této práce vytvořit nad daty strojové učení.

SPARQL Protocol and RDF Query Language (SPARQL) se velice podobá jazykům pro relační databáze, ovšem je upravený pro dotazy nad RDF trojicemi. Každý prvek z trojice může být nahrazený proměnnou, která pak bude vypsaná ve výsledku. Podobnost s relačními databázemi je hlavně použití stejných příkazů jako *SELECT*, *FROM* nebo *WHERE*. Kromě dotazování lze také pomocí tohoto jazyka trojice tvořit, a to příkazem *CONSTRUCT*, který nejen že vyhledá data podle zadaných kritérií (jako *SELECT*), ale data transformuje do předepsané podoby a vypíše ve formátu trojic.

V představení RDF také nesmí být opomenut způsob uložení dat, neboli jeho formáty. RDF nabízí k reprezentaci dat více serializačních formátů, které dodržují základ trojic, od sebe se ale liší různými vlastnostmi, například v propojení s jinými jazyky jako XML[4] nebo JSON[5], možností vytváření grafů nebo jednoduchostí. Mezi tyto formáty patří:

- RDF/XML[6] – Formát, který definuje XML syntax k popisu RDF grafu.
- JSON-LD[7] – *JavaScript Object Notation for Linked Data* je jednoduchý syntaktický formát definující serializaci propojených dat (Linked Data) do formátu JSON.
- RDFa[8] – *Resource Description Framework in Attributes* je technika, která dovoluje vkládat strukturovaná (RDF) data do HTML dokumentů.

[2]Více informací online: `https://www.w3.org/` (13. 3. 2016).
[3]URI – Uniform Resource Identifier – Strukturovaný textový řetězec ke specifikaci zdroje.
[4]XML – Extensible Markup Language.
[5]JSON – JavaScript Object Notation.
[6]Specifikace online: `https://www.w3.org/TR/rdf-syntax-grammar/` (26. 4. 2016).
[7]Specifikace online: `https://www.w3.org/TR/json-ld/` (26. 4. 2016).
[8]Specifikace online: `https://www.w3.org/TR/xhtml-rdfa-primer/` (26. 4. 2016).

- Turtle[9] – *Terse RDF Triple Language* dovoluje vytvořit jednoduchý kompaktní textový dokument popisující RDF graf.
- N-Triples[10] – Jednoduchý řádkový formát pro RDF grafy, který je lehce rozložitelný (*parse*) a tvoří podmnožinu jazyka Turtle.
- N-Quads[11] – Je rozšířením formátu N-Triples, které dovoluje použití více RDF grafů najednou.
- N3[12] – *Notation3* je formát podobný Turtle, který je rozšířen o další vlastnosti.
- TRiG[13] – Je rozšířením formátu Turtle, které dovoluje použití více RDF grafů najednou.

Součástí některých formátů je i používání takzvaných prefixů, které slouží ke zkrácení a zpřehlednění identifikátorů RDF trojic. V praxi to znamená, že je na začátku RDF souboru definován prefix například pro DBpedia ontologie tak, že štítku „dbo:“ je přiřazena URI odkazující na prostor ontologie „http://dbpedia.org/ontology/“[14]. Poté v reprezentaci trojic může být použita třída z DBpedia ontologie vypsáním štítku před požadovanou třídou (například „dbo:Place“).

1.1.2 Linked Data

Princip propojených dat (Linked Data) představil v roce 2006 Tim Bernes Lee [3]. Linked Data pojednávají o množině doporučených postupů pro publikování strukturovaných dat na webu. Motivací pro dodržování těchto postupů je vytvoření souboru propojených dat popisujících objekty různými zdroji. Principy Linked Data jsou:

1. Používat URI identifikátory za názvy objektů.
2. Používat HTTP[15] URI, aby lidé tyto objekty mohli nalézt.
3. Používat čitelné URI (člověk bude schopný pohledem na URI zjistit užitečné informace).
4. Vkládat odkazy na jiné URI identifikátory, aby bylo možné objevit více objektů.

[9] Specifikace online: `https://www.w3.org/TR/turtle/` (26. 4. 2016).
[10] Specifikace online: `https://www.w3.org/TR/n-triples/` (26. 4. 2016).
[11] Specifikace online: `https://www.w3.org/TR/n-quads/` (26. 4. 2016).
[12] Specifikace online: `https://www.w3.org/TeamSubmission/n3/` (26. 4. 2016).
[13] Specifikace online: `https://www.w3.org/TR/trig/` (26. 4. 2016).
[14] Celý příkaz: @prefix dbo: http://dbpedia.org/ontology/.
[15] HTTP – Hypertext Transfer Protocol – Aplikační protokol sloužící k výměně HTML dokumentů.

Součástí principů je i používat standardy, jakým je například RDF z oddílu 1.1.1, k reprezentaci a přístupu k datům. Na základě takto definovaných propojených dat vznikla komunita Linking Open Data (LOD)[16], jejíž cílem je rozšířit stávající web o informace, které jsou publikovány pod otevřenou licencí. Ovšem ve formě Linked Data principů, to znamená ve formátu RDF, a propojená odkazy mezi sebou. Příkladem dat zveřejněných pod otevřenou licencí jsou informace například ze serverů Wikipedia[17], Geonames[18], WordNet[19], atd. V důsledku vytvoření takovéto databáze propojených dat vypracoval Linking Open Data projekt graf spojení jednotlivých datových sad a tento graf zveřejnil jako interaktivní vizualizaci, kterou nazval The Linking Open Data cloud diagram[20]. V grafu propojení datasetů[21] je znázorněna i přibližná velikost, přesněji porovnání velikost datasetů. Příklad je vidět na obrázku 1.1, ze kterého lze vyčíst, že jedním z největších datasetů propojených dat je DBpedia.

1.2 Strojové učení

1.2.1 Popis

Machine learning, česky strojové učení, je součástí počítačové vědy a pochází ze studií zabývajících se rozpoznáváním vzorů (*pattern recognition*) a výpočetním učením (*computational learning*) spadajících pod umělou inteligenci (*artificial intelligence*). Strojové učení se zabývá technikami a algoritmy, které umožňují počítačovým systémům schopnost učit se. V této problematice chápeme učení se jako změnu stavu systému, která umožní lepší přizpůsobení ke změnám okolního prostředí. Hlavním zdrojem k vytvoření popisu strojového učení byl článek prof. Berky [4].

Učení lze rozdělit na dva typy: učení se znalostem (*knowledge aquisition*) a učení se dovednostem (*skill refinement*). Tato práce se bude zabývat hlavně prvním typem, protože odpovídá učení, které lze aplikovat na pojmenované entity a jejich domény. Znalostní učení v první fázi svého procesu hledá pravidla a ontologie, která jsou extrahována z trénovacích dat. Ve druhé fázi tato pravidla použije k určování a předpovědi událostí na základě nově předložených dat. V tomto případě se bude jednat o extrakci vlastností trénovacích entit a vytvoření modelů pro predikci domény uživatelem zadané entity.

Dalším aspektem kategorizace učení je rozdělení podle zjištění informace, že se systém učí správně. Strojové učení lze rozdělit do kategorií: učení s učite-

[16] Více informací online: `https://www.w3.org/wiki/SweoIG/TaskForces/CommunityProjects/LinkingOpenData/` (14. 3. 2016).

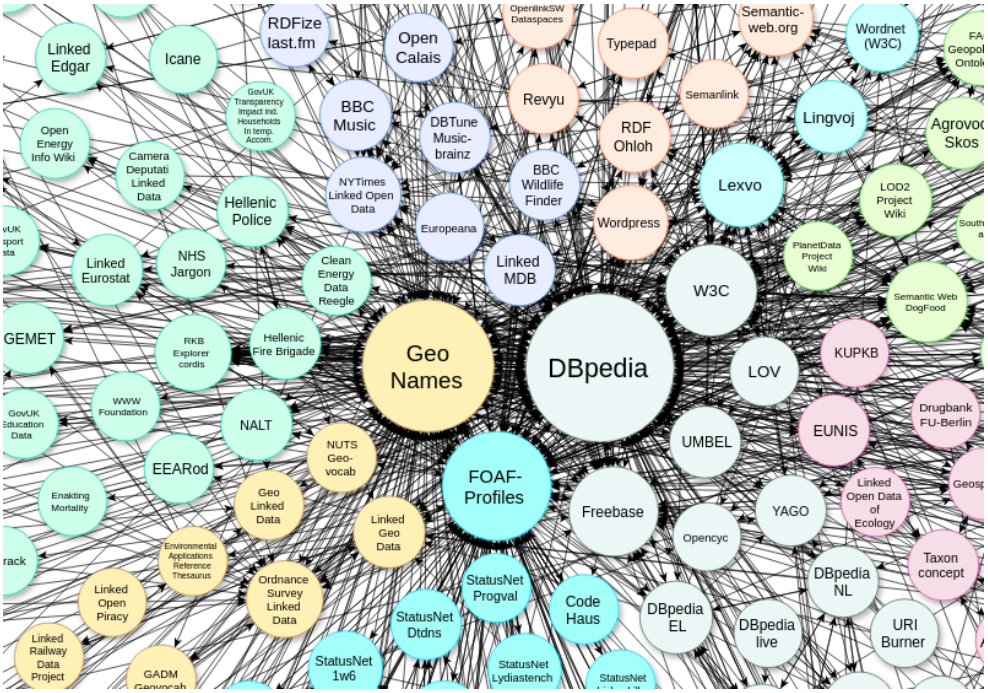
[17] Více informací online: `http://www.wikipedia.org/` (14. 3. 2016).

[18] Více informací online: `http://www.geonames.org/` (14. 3. 2016).

[19] Více informací online: `http://wordnet.princeton.edu/online/` (14. 3. 2016).

[20] Více informací online: `http://lod-cloud.net/` (14. 3. 2016).

[21] Dataset – Zdroj strukturovaných dat.



Obrázek 1.1: Ukázka vizualizace propojení dat. Výseč digramu vytvořený komunitou Linking Open Data. Pořízeno 7. 3. 2016 jako snímek obrazovky webové stránky `http://lod-cloud.net/versions/2014-08-30/lod-cloud.svg`.

lem (*supervised learning*), učení bez učitele (*unsupervised learning*) a zpětnovazební učení (*reinforcement learning*). Také je rozlišováno učení podle reprezentace dat v procesu učení na atributové a relační, podle způsobu zpracování na dávkové a inkrementální a podle formy učení na empirické a analytické. Učení domén pojmenovaných entit lze zařadit mezi učení s učitelem, jehož atributová reprezentace dat je zpracovávána dávkově a formou empirického učení. Význam jednotlivých kategorií bude více vysvětlen v následujících odstavcích.

Učení s učitelem je založeno na zpracování trénovacích dat, které mají pro své vstupy určené jasné výstupy (dané učitelem). Cílem učení je vytvořit model pravidel, které ze vstupu zvolí výstup. S tímto kontextem úzce souvisí volba reprezentace dat. Pod pojmem reprezentace dat atributy si lze představit datovou tabulku s vlastnostmi objektů v jednotlivých řádcích. Tato dvě zařazení lze z pohledu této práce chápat tak, že formát trénovacích entit bude tvořen vlastnostmi entit (atributy) a přesně danými doménami (třídami).

Úzce spjaté s učením s učitelem je empirické učení, které pracuje s velkým množstvím příkladů a pomocí rozhodovací funkce hledá znalosti, které pro hodnoty vstupních atributů určí výstupní atributy. Tento způsob zpracování

dat bude v této práci probíhat dávkově, což znamená, že vytvoření modelu na trénovacích datech proběhne vždy najednou (zpracování nebude probíhat postupně, inkrementálně). To je dáno faktem, že domény i pojmenované entity se často nemění a nepřibývají, tím pádem není potřeba inkrementálně přeučovat nebo doučovat již vytvořené modely.

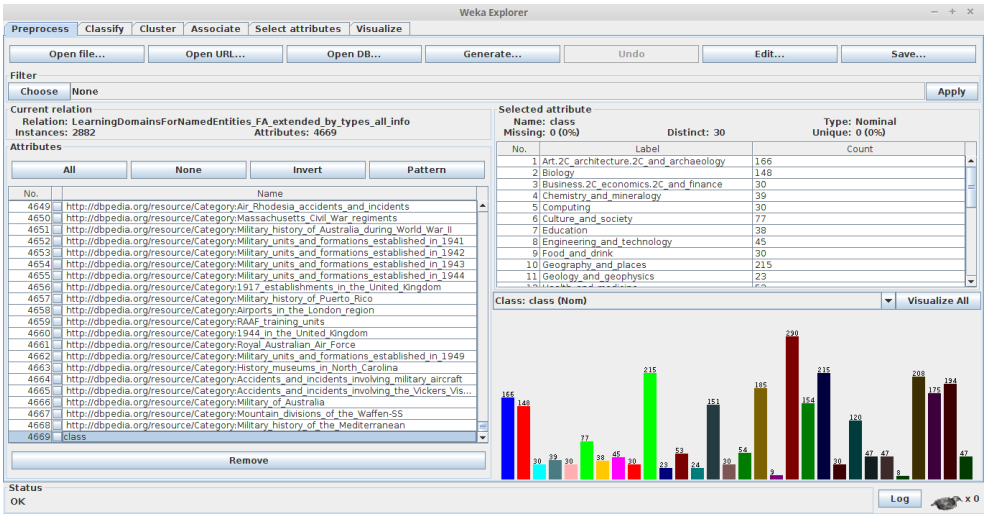
1.2.2 Nástroje

Existuje mnoho softwarových nástrojů určených ke strojovému učení. Tyto programy, většinou nabízející grafické uživatelské prostředí, se skládají z mnoha učících algoritmů, data pre-procesorů, klasifikátorů a nástrojů pro experimenty a vizualizaci výsledků. Pro práci se strojovým učením byl vedoucím práce doporučen program Weka.

Waikato Environment for Knowledge Analysis (Weka)[22] vyvinutý University of Waikato se skládá z kolekce algoritmů určených k strojovému učení a je napsaný v jazyce Java. Software je distribuovaný pod licencí GNU GPL[23] a nabízí jak program s grafickým uživatelským rozhraním, tak i Java knihovny určené k integraci do dalších aplikací. V této práci byly využity obě varianty: grafická k vygenerování natrénovaných modelů a knihovny k predikování domény pomocí REST[24] služby. V kontextu práce s daty Weka používá Attribute-Relation File Format (ARFF)[25]. V tomto souboru jsou data uložena jako seznam instancí sdílejících množinu atributů. V případě této práce jsou instance tvořeny trénovacími entitami a množinou atributů jsou jejich vlastnosti. Soubor je rozdělen na dvě části: atributy a data. V první části jsou definovány všechny atributy. Každý atribut může nabývat buď hodnoty *t*, nebo *f* (vyskytuje se nebo nevyskytuje pro danou entitu), výjimkou je atribut *class* (třída), který je tvořen názvy všech domén. V části s daty jsou vypsány samotné trénovací instance, které jsou vyjádřeny řádkem skládajícím se z písmen *t* nebo *f*, oddělených čárkou a zakončeny názvem třídy, do které patří. Takto strukturovaný soubor je program Weka schopný zpracovat a pomocí různých algoritmů natrénovat model k dalšímu použití. Náhled grafického rozhraní nástroje Weka lze vidět na obrázku 1.2.

Dalším nástrojem, který je nutné v tomto kontextu zmínit, protože byl použit v projektu z podkapitoly 1.3, je program RapidMiner[26]. Jedná se o komerční software s open-source[27] verzemi, napsaný v jazyce Java na katedře Umělé inteligence Technical University of Dortmund. Nástroj RapidMiner for-

[22] Více informací online: `http://www.cs.waikato.ac.nz/ml/weka/` (18. 3. 2016).
[23] Více informací online: `http://www.gnu.org/` (18. 3. 2016).
[24] REST – Representational State Transfer.
[25] Více informací online: `http://www.cs.waikato.ac.nz/ml/weka/arff.html` (18. 3. 2016).
[26] Více informací online: `http://rapidminer.com/` (18. 3. 2016).
[27] Open-source – Definice ze serveru dictionary.cambridge.org „Open-source software is free to use, and the original program can be changed by anyone.“ [5].



Obrázek 1.2: Ukázka grafického rozhraní nástroje Weka Explorer. Pořízeno 12. 4. 2016 jako snímek obrazovky spuštěného programu.

mou grafického uživatelského rozhraní nebo využitím API[28] slouží ke strojovému učení, dolování dat a různým predikčním a business analytikám.

1.2.3 Klasifikace

Proces učení modelů pomocí trénovacích dat, učení s učitelem, se nazývá klasifikace. Je to statistický proces, který identifikuje kategorii nové instance dat na základě dat trénovacích, u kterých je jejich kategorie známá. Algoritmům, které implementují proces klasifikace, se také říká klasifikátory. Klasifikátory lze rozdělit do několika skupin podle technik, které používají, na: Logické algoritmy, Perceptronové sítě, Statistické algoritmy, Instanční algoritmy a Support Vector Machines (SVM).

1.2.4 Evaluační metody a metriky

Z předchozí části 1.2.3 víme, že klasifikace natrénuje a vytvoří modely, které se dále používají k určování neznámých tříd instancí pomocí trénovacích dat. Ovšem různé klasifikátory založené na odlišných algoritmech se mohou hodit pro různé typy dat, proto po vytvoření modelu klasifikátorem následuje proces evaluace.

Evaluace je technika ohodnocení výkonu, přesněji účinnosti klasifikátoru. Používají se různé metody, které jsou ovšem založené na vyhodnocení datasetu pomocí dat trénovacích a testovacích. Proces evaluace je součástí klasifikátoru, který nejdříve natrénuje model podle zvoleného algoritmu a poté

[28] API – Application Programming Interface – Rozhraní pro programování aplikací.

na trénovacích datech vyhodnotí jeho úspěšnost. Mezi techniky evaluace nebo také testování patří:

- **Použití stejného datasetu** – tím se rozumí použití stejných dat pro trénování i testování.
- **Různé datové sady** – testování probíhá na úplně jiném datasetu, ve většině případů vytvořeným jinou technikou nebo ručně.
- **Cross-validation** – používá se pouze jeden dataset, který se rozdělí na více částí, například N, poté jednu část použije jako testovací a tento proces opakuje Nx. Výsledkem jsou pak zprůměrované výstupy jednotlivých opakování.
- **Procentuální rozdělení** – data se rozdělí podle zadaných procent na trénovací a testovací, například 66% trénovací data a 34% testovací data.

Z principu už je jasné, že proces evaluace musí mít nějaký výstup, podle kterého se natrénované modely dají porovnávat. Tento výstup se skládá z metrik, které nástroj Weka z oddílu 1.2.2, rozděluje na dvě skupiny, první popisuje celý model a druhá popisuje klasifikační třídy. V následujících podsekcích budou představeny jednotlivé metriky se zachováním anglického názvu, které pak budou více vysvětleny.

1.2.4.1 Metriky popisující celý model

- **Correctly Classified Instances** – Správně klasifikované instance. Počet instancí, kterým predikce přiřadila odpovídající třídu.
- **Incorrectly Classified Instances** – Špatně klasifikované instance. Počet instancí, kterým predikce přiřadila jinou třídu.
- **Kappa statistic** – Shoda predikce se správnými třídami.
- **Mean absolute error** – Průměrná absolutní chyba. Průměrná velikost chyb jednotlivých predikcí.
- **Root mean squared error** – Kvadratická průměrná chyba. Průměrná velikost chyb jednotlivých predikcí, které jsou ovšem umocněny před průměrováním. Je užitečná při výskytu velkých chyb.
- **Relative absolute error** – Relativní absolutní chyba. Průměrná absolutní chyba, která je normalizovaná celkovou chybou predikování.
- **Root relative squared error** – Kvadratická relativní absolutní chyba. Stejná jako relativní absolutní chyba, ovšem umocněná před průměrováním.
- **Total Number of Instances** – Celkový počet instancí.

1.2.4.2 Metriky popisující jednotlivé klasifikační třídy

- **TP Rate** – True positive rate – Procentuální zastoupení správných predikcí třídy.
- **FP Rate** – False positive rate – Procentuální zastoupení nesprávných predikcí třídy, tzn. třída byla zvolena, ale měla být zvolená jiná.
- **Precision** – Přesnost. Pro třídu x je to počet správných predikcí x ku všem predikcím x.
- **Recall** – Citlivost. Pro třídu x je to počet správných predikcí x ku počtu všech x. Stejné jako TP Rate.
- **F-measure** – Míra testované přesnosti použitelná k porovnání klasifikátorů. Je nazývána také jako harmonický průměr mezi *Precision* a *Recall*. $F\text{-}measure = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall}$.
- **ROC Area** – Zkratka pro Receiver operating characteristic, konkrétně pro oblast pod grafem. Vyjadřuje pravděpodobnost, že počet náhodně vybraných instancí predikovaných klasifikátorem správně bude vyšší než počet náhodně vybraných instancí predikovaných špatně.
- **Class** – Třída, která má být predikována.

Hlavními metrikami, které budou zajímavé v kontextu této práce pro porovnání modelů jsou Míra testované přesnosti (*F-measure*) a s ní spojená přesnost (*Precision*) a citlivost (*Recall*).

1.3 Dosavadní řešení

Jedním z námětů na realizaci této práce byl projekt *DBpedia Domains: augmenting DBpedia with domain information* [1]. Jedná se o první realizaci automatického přiřazení DBpedia domén pomocí Wikipedia kategorií technikou *kernel-based k-means clustering*. Tuto práci a tento projekt tedy spojuje jeden cíl, učení domén pojmenovaných entit, konkrétně entit DBpedie, ovšem svá učení staví na různých technikách.

Strojové učení, které zvolil tým projektu DBpedia Domains, spadá do kategorie učení bez učitele. Tento typ učení se úzce spojuje se statistickým problémem odhadu hustoty (*density estimation*), ale také zahrnuje mnoho dalších technik, které se snaží shrnout a vysvětlit klíčové vlastnosti dat. Jednou z technik učení bez učitele je i klastrování (*clustering*), jehož cílem je seskupit množinu objektů do skupin, ve kterých budou objekty mezi sebou nejvíce podobné.

Kernel-based k-means clustering, což lze přeložit jako jádrově založené klastrování s k-sousedy, se řadí mezi jednu z metod klastrování. Tato technika funguje stejně jako algoritmus Nejbližších sousedů, rozdílem je nahrazení

Eukleidovské vzdálenosti za nějakou jádrovou metodu. Autoři vzali v úvahu 4 jádrové funkce, které pak mezi sebou porovnávali. Za tyto funkce zvolili:

- **Jednoduchou podobnostní funkci** zaznamenávající fakt, že pokud mají dvě kategorie tendenci být přiřazeny ke stejným stránkám, tak jsou si podobnější.
- **Více distribučních funkcí** porovnávajících pravděpodobnostní distribuční funkce kategorií podle jejich stupně podobnosti. Za pravděpodobnostní funkce byly zvoleny Eukleidovy kvadratické vzdálenosti L_1 a L_2, Jensen-Shannon divergence a Hellingerova vzdálenost.
- **Funkci založenou na porovnání řetězců**, která porovnává názvy kategorií.
- **Stromovou funkci**, která vypočítává podobnost jako stupeň překrytí stromů kategorií do nějaké maximální hloubky.

Systém DBpedia Domains byl implementován pomocí frameworku nabízeným open-source strojově učícím programem RapidMiner z oddílu 1.2.2. Pro vyhodnocení svých výsledků pak autoři použili články z Wikipeadia Featured Articles, ze kterých náhodně vybrali 60 vzorků (2 z každé kategorie) a otestovali, zda doména určená jejich funkcemi odpovídá kategorii, do které spadá článek z Featured Articles.

Výsledkem práce projektu DBpedia Domains bylo porovnání všech jádrových funkcí pomocí různých metrik, jejichž součástí byla i *F-measure* z oddílu 1.2.4. Kromě toho také přidali k porovnání i neváženou lineární kombinaci všech použitých jádrových funkcí, která se ve výsledku ukázala jako nejlepší možnost s *F-measure* = 56.75%. V závěrečné diskuzi autoři popisují výsledky jednotlivých měření a různá spojení kategorií do větších celků. Také hodnotí celou práci tak, že z pohledu výsledků je úkol proveditelný, ovšem z hlediska úrovně blízké člověku, kterou má DBpedia, je stále na čem pracovat. V návaznosti na to hovoří o práci do budoucnosti, kterou by mohl být stejný úkol, ale postavený na supervizovaném učení, což je motivací této práce, která se přesně tímto problémem zabývá.

Dalším projektem vztahující se k této práci je *Type Inference on Noisy RDF Data* [6], který se může zdát jako nesouvisející s tématem této práce, ale opak je pravdou, protože jeho základem je odvození typu entity za pomoci jejich predikátů. Cílem této práce je doplnění nebo určení typu entity na základě různých RDF informacích o ní, mezi které patří i predikáty. Pro svojí potřebu autoři zavádí pojem *SDType*, jehož název vychází z pravděpodobnostního rozdělení (*Statistical Distribudion*), které je použito pro predikci typu instancí. *SDType* využívá odkazy mezi instancemi k určení jejich typu s použitím vážené volby. Projekt pracuje s datasety DBpedia a OpenCyc, které používá ke zhodnocení svého přístupu k problému. O těchto datasetech

vytvořil zajímavou charakteristiku z pohledu počtu instancí, počtu odlišných tříd a vlastností, průměrného počtu typů pro instanci nebo průměrného počtu vstupních a výstupních vlastností instancí.

Ve svých závěrech autoři hodnotí práci jako úspěšnou. Naměřili *F-measure* 88.9% pro DBpedii a 63.7% pro OpenCyc. Dále pak popisují, že *SDType* dobře predikuje obecné třídy jako například *Band*[29], ale už je pro něj složitější predikovat typ u více specifičtějších tříd jako například *Heavy Metal Band*[30].

[29]Band – česky: hudební skupina.

[30]Heavy Metal Band – česky: heavy metalová hudební skupina.

Kapitola 2

Analýza

V této kapitole bude analyzován problém učení domén pojmenovaných entit, přiblíženo zadání a požadavky na práci, analyzována trénovací data a jejich zdroje, popsána analýza klasifikátorů a představena REST služba.

2.1 Problém učení domén pojmenovaných entit z obecného hlediska

V předchozích kapitolách bylo představeno strojové učení 1.2 a sémantický web 1.1.1. Tato část se bude zabývat propojením těchto dvou pojmů, které budou aplikovány do kontextu práce s pojmenovanými entitami.

2.1.1 Domény

Součástí práce je taktéž vhodně zvolit rozdělení domén, do kterých pak budou zařazovány pojmenované entity. Nejprve bude více přiblížen pojem doména. Z definice podle serveru dictionary.cambridge.org [7] je doména chápána jako oblast zájmu nebo oblast, nad kterou má člověk kontrolu. To lze v kontextu pojmenovaných entit vysvětlit jako široký okruh nebo zařazení, do kterého spadají. Jednoduchým příkladem může být entita „Lední hokej“, kterou lze zařadit do domény „Sport a rekreace“. Takovéto zařazení se může zdát přirozené, ovšem ze zachování ucelené formy je nutné určit hlavní domény, se kterými bude systém pracovat.

Jedním ze zdrojů domén určeným ke klasifikaci je projekt TaaS[31]. Tento projekt je dotovaný Evropskou unií a jeho motivací je efektivní řešení pro práci s terminologií, okamžitý přístup k aktuálním termům a znovu použitelnost zdrojů. Ke své práci s terminologií TaaS vytvořil specifikaci domén[32], která je použitelná pro pojmenované entity.

[31] Více informací online: `https://term.tilde.com/taas/` (16. 3. 2016).
[32] Více informací online: `https://term.tilde.com/domains/` (16. 3. 2016).

Za další použitelný zdroj ke specifikování domén lze použít Wikipedia Featured Articles[33]. Jak už název napovídá, jedná se o články z Wikipedie představené v sekci 2.4. Podle editorů se tyto články považují za nejlepší, které Wikipedia může nabídnout, a jsou dávány za příklad pro vytváření dalších článků. Vzhledem k pravidlům a procesu zařazení mezi Featured Articles se dají tyto články považovat za důvěryhodné, a proto i jejich rozřazení do oblastí zájmu se dá považovat za specifikaci domén. K březnu 2016 obsahují 4 727 článků, které jsou rozřazeny do 30 kategorií (oblastí zájmu). Některé z těchto kategorií jsou dále rozděleny i na podkategorie, ukázkou může být například kategorie „Biology“ s podkategoriemi „Animals“ a „Biology biographies“.

2.1.2 Pojmenované entity

Pojmenovanými entitami jsou podle publikace *Zpracování pojmenovaných entit v českých textech* od M. Ševčíkové, Z. Žabokrtského a O. Krůzy: „[...] slova a slovní spojení, která v textu vystupují jako jména osob, geografické názvy, jména produktů, názvy organizací, ale také jako časové údaje apod. [...]“ [8, str. 5]. Tato definice člověku přímo nabízí mít možnost rozdělovat pojmenované entity do domén už jen proto, že většina lidí se při čtení textů zajímá jen o určité téma (doménu), nebo při vyhledávání textových dokumentů může být uživateli nabízen text jen z domény, o kterou má zájem.

Samozřejmě existuje řada nástrojů na rozpoznání pojmenovaných entit (NER[34]), ovšem ty pro daný text pouze klasifikují pojmenované entity, ale neurčí, do jaké domény spadají. Něco jiného by bylo, pokud by textové dokumenty obsahovaly i význam textu, například obdobně jako u sémantického webu z oddílu 1.1.1. Z toho by se dal pro jednotlivé entity extrahovat jejich význam a dále určit zařazení nebo dokonce doména, do které spadají. Tato myšlenka tuto práci vede k zatím největšímu zdroji volného textu a pojmenovaných entit na internetu, a to k Wikipedii. Z popisu v podkapitole 2.4 se dozvíme, že každá entita Wikipedie je popsána jako zdroj v datasetu DBpedie. Ale i když Wikipedia při vytváření popisu entity nabízí uživateli možnost vyplnění kategorie, do které entita spadá, nechává mu při zadávání volnou ruku. Proto se stává, že buď entita žádnou kategorii nastavenou nemá, anebo spadá do jedné (nebo více) z přibližně 300 tisíc kategorií, což nesplňuje myšlenku rozdělení do domén z definice z oddílu 2.1.1. A právě tento problém byl motivací projektu DBpedia Domains z podkapitoly 1.3 a je také motivací této práce.

Toto spojení pojmenovaných entit a domén lze kvůli struktuře datasetů sémantického webu doplnit o strojové učení. Jak už víme z podkapitoly 1.2, existuje mnoho druhů učení, tato práce se ale zaměří na učení s učitelem. Díky strukturovaným datům jsou informace o entitách dobře strojově rozpo-

[33] Více informací online: `https://en.wikipedia.org/wiki/Wikipedia:Featured_articles` (16. 3. 2016).

[34] NER – Named Entity Recognition.

znatelné, tudíž může být na nějakém vzorku dat podle zvolených parametrů nebo vlastností natrénovaný model, který poté bude predikovat libovolné pojmenované entitě její doménu.

Myšlenka sjednocení těchto tří odvětví a vytvoření nástroje využitelného například k lepšímu rozřazení článků Wikipedie, k doporučování článků nebo lepšímu vyhledávání v nich, je důvodem sepsání této diplomové práce.

2.2 Zadání a požadavky

Zadáním této práce je přímo se zaměřit na problém učení domén pojmenovaných entit. To znamená, že je nutné se zpočátku seznámit s aktuálním stavem tohoto problému, tedy analyzovat aktuální řešení a provést rešerši. S tímto seznámením souvisí analýzy existujících datasetů a metod strojového učení, na jejichž základech by se mělo rozhodnout, jak bude navržen a implementován nástroj na generování trénovacích datasetů. Z těchto datasetů poté budou vytvořeny modely, které budou použity pro automatické učení domén entit. Zadáním této práce je taktéž vytvořit rozhraní v podobě REST služeb, které bude sloužit právě jako hlavní nástroj automatického učení domén pojmenovaných entit a bude založené na vytvořených modelech. V neposlední řadě je nutné nad těmito modely provést experimenty.

Z popisu tohoto zadání vyplývají formální funkční a nefunkční požadavky výsledných nástrojů, které zde budou představeny.

2.2.1 Funkční požadavky

F1. Systém bude generovat trénovací datasety ve formátu ARFF.

F2. Systém bude konfigurovatelný pomocí vstupních argumentů nebo konfiguračních souborů.

F3. Systém bude obsahovat pomocné funkce k extrakci domén a analýze vlastností entit.

F4. Systém bude schopen přijmout od uživatele identifikátor pojmenované entity.

F5. Systém provede analýzu pojmenované entity na základě přijatého identifikátoru.

F6. S využitím analýzy pojmenované entity systém provede automatické učení její domény pomocí natrénovaných modelů.

F7. Systém předá výslednou doménu pojmenované entity uživateli ve formátu JSON nebo *text/plain*.

2.2.2 Nefunkční požadavky

N1. Výsledný systém bude napsaný v jazyce Java.

N2. Rozhraní pro automatické učení domén pojmenovaných entit bude implementováno jako REST služba.

N3. Systém bude rozdělen na 2 samostatné aplikace:

- Jedna aplikace bude vytvářet trénovací datasety,
- druhá aplikace bude tvořit rozhraní REST služeb pro automatické učení domén entit.

N4. Systém bude navržený tak, aby byl použitelný pro různé zdroje pojmenovaných entit ve formátu HDT.

2.3 Analýza existujících datasetů

Úspěšným základem učení s učitelem je vhodný výběr trénovacích dat. Tato data by měla mít nejen co nejširší zastoupení unikátních informací, ale také možnost rozdělení do větších celků nebo přímo domén. Dalším aspektem, který bude v této podkapitole analyzován, je význam pojmu dataset, a to pomocí kritéria, jaká data obsahuje.

Vhodným zdrojem a přehledem existujících datasetů je The Linking Open Data cloud diagram (dále jen LOD cloud) popsaný v oddíle 1.1.2. Jak už bylo popsáno, LOD cloud obsahuje všechny volně dostupné datasety a zobrazuje jejich vzájemné projení. Další vlastností tohoto diagramu je vizualizace přibližné velikosti jednotlivých datasetů oproti ostatním datasetům a také barevné rozlišení datasetů podle odvětví, do kterého spadají. LOD cloud bere v úvahu následující odvětví:

- *Publications* – Zde se vyskytují datasety obsahující data především o publikacích, tedy datasety knihoven a nakladatelství.
- *Life Sciences* – Datasety spadající do této skupiny obsahují především biologická data.
- *Social Networking* – Tato skupina obsahuje převážně statistické datasety instancí serveru *GNU social*[35] (dříve StatusNet). Jedná se tedy o data získaná ze sociálních sítí nebo informace o uživatelských profilech (zejména používající slovník *foaf*[36])
- *Geographic* – Do tohoto odvětví spadají data týkající se zeměpisných dat, tedy datasety obsahující názvy zemí, měst, poštovních směrovacích čísel, ale hlavně obsahující informace o zeměpisných souřadnicích.

[35] Více informací online: `https://gnu.io/social/` (27. 4. 2016).
[36] Více informací online: `http://www.foaf-project.org/` (27. 4. 2016).

- *Government* – V této skupině se vyskytují data státních orgánů, například datasety vládních institucí Velké Británie nebo datasety Evropského patentového úřadu.

- *Media* – Datasety spadající do této skupiny obsahují především data televizí a rádií.

- *User-Generated Content* – Toto odvětví je zaměřeno na data generovaná uživateli. Například dataset *Flickr Wrappr*, který se snaží o propojení informací s fotografiemi od uživatelů.

- *Linguistics* – Do tohoto odvětví spadají datasety obsahující informace o přirozeném jazyku.

- *Cross-Domain* – Tato skupina obsahuje datasety všeobecné, tedy obsahující data ze všech odvětví.

Z těchto odvětví bude pro potřeby této práce nejvhodnější vybírat dataset z posledního jmenovaného odvětví (*Cross-Domain*). Důvodem je potřeba mít k dispozici co nejobecnější a nejrůznorodější data, aby jejich dělení do domén bylo relevantní.

Výběr datasetu bude taktéž závislý na jeho velikosti oproti ostatním datasetům. Je předpokládáno, že čím je dataset větší, tím více unikátních entit obsahuje. Výrazněji většími datasety v LOD cloudu jsou DBpedia, GeoNames a FOAF-Profiles (lze vidět na obrázku 1.1). Z těchto tří datasetů je pro tuto práci nejhodnější dataset DBpedia, protože spadá do odvětví *Cross-Domain*. Ostatní dva spadají do skupin *Geographic* a *Social Networking*, proto nejsou k učení domén entit příliš vhodné. Další výhodou datasetu DBpedia je to, že obsahuje velký počet propojení s jinými datasety přes predikát *owl:sameAs*[37], a proto je možné logiku vybudovanou nad tímto datasetem snadno převést na jiný dataset právě přes tato propojení. Více bude dataset DBpedia představen v podkapitole 2.4.

Dataset DBpedia je vhodný k použití i svou připraveností k rozdělení do kategorií, a to konkrétně pomocí Wikipedia Featured Articles z oddílu 2.1.1. Featured Articles jsou Wikipedia články rozdělené do kategorií širokého významu, které můžeme považovat za domény, a tudíž díky tomu, že Wikipedia články jsou reprezentovány jako zdroje v DBpedia, můžeme Featured Articles považovat za dobrý základ pro trénovací instance. Pokud by se ovšem vycházelo z nějakého standardu, například TaaS z oddílu 2.1.1, musel by být jednotlivým doménám ručně vytvořen seznam entit, které je reprezentují. Tento postup je však časově velmi náročný a navíc by mohl být výběr entit jedním člověkem subjektivně zkreslený. Oproti tomu Featured Articles jsou tvořeny crowd-sourcingově, tedy celou komunitou lidí.

[37] Více informací online: `https://www.w3.org/TR/owl-ref/#sameAs-def` (27. 4. 2016).

Takto byla provedena analýza a výběr zdroje pojmenovaných entit a zdroje trénovacích dat. Vybraný formát datasetů není vhodný pro přímé trénování, protože pokud chceme použít klasifikaci, je nutné aby každá instance trénovacích dat (entita) byla tvořena atributy společnými pro všechny instance, pouze s příznaky, zda entita obsahuje atribut, nebo ho neobsahuje. Podle tohoto kritéria je pro naše účely možné datasety dělit na dvě skupiny: jako zdroj pojmenovaných entit a zdroj entit ve formátu vhodném k trénování. Pro první skupinu byly použity datasety DBpedia a druhá skupina datasetů musela být vytvořena.

Samotné vytvoření datasetů by obnášelo pouze načtení entit Featured Articles a jejich převedení na dataset trénovací. Ale při bližším pohledu na entity Featured Articles a jejich RDF trojice si můžeme všimnout, že jsou entity tvořeny zčásti společnými (obecnými) informacemi a z části hodně specifickými informacemi. Proto je nutné vybrat vlastnosti entit, podle kterých bude probíhat učení.

Ideálními vlastnostmi jsou vlastnosti, které jsou společné pro všechny entity a obsahují strukturované informace, nejlépe referenci na jiné entity nebo objekty. Tudíž máme-li entity tvořené RDF trojicemi, jejich vlastnosti identifikujeme predikáty a atributy budou tvořit objekty vztahující se k nim. Příkladem mohou být predikáty *dbo:abstract* nebo *dbp:caption*, které obsahují všechny entity, ale jejich objektem je volný text, který se nehodí k jednoznačnému učení nebo by bylo nutné použít další nástroje k jeho zpracování. Vhodnými predikáty jsou:

- **dct:subjekt**[38] – obsahuje kategorie, které jsou zobrazeny pod Wikipedia článkem a které nastavují sami editoři. Pro tyto kategorie platí pravidla jako jmenné konvence nebo stromová struktura, ale nedají se použít jako domény z důvodů, které byly vysvětleny v podkategorii 2.1.

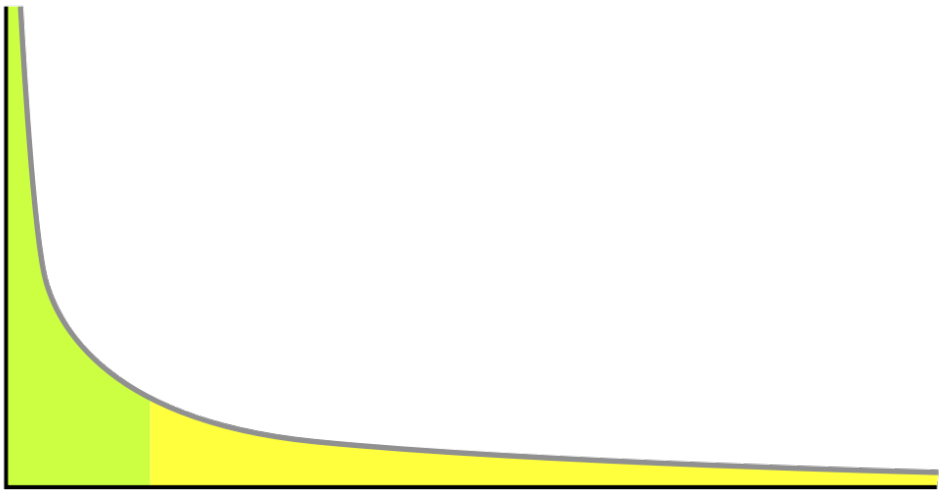
- **rdf:type**[39] – se skládá z tříd, které představují entitu. Tyto třídy se extrahují z Wikipedia infoboxů, které obsahují základní data a jsou vyplňovány editory článku.

Na daný problém lze nahlížet i z jiného úhlu pohledu. Nemusíme se zaměřit přímo na konkrétní predikát, ale na všechny predikáty dané entity jako na celek. Myšlenkou učení na základě predikátů je, že entity ze stejné domény jsou tvořeny podobnou množinou predikátů jen s různými objekty. Například entity z domény *Sport* můžou obsahovat týmy, sportovce, sportovní akce nebo ligy.

V kontextu této práce bylo provedeno rozhodnutí, že trénovací datasety budou postaveny na třech výše popsaných vlastnostech entit. S použitím různých klasifikátorů z další podkapitoly pro ně budou vytvořeny samostatné

[38] *dct:subjekt* – Identifikátor v úplném tvaru: `http://purl.org/dc/terms/subject`.

[39] *rdf:type* – Identifikátor v úplném tvaru: `https://www.w3.org/1999/02/22-rdf-syntax-ns#type`.



Obrázek 2.1: Graf Dlouhého chvostu (zdroj Wikipedia).

modely a jeden společný (sjednocením atributů), které mezi sebou budou porovnány a budou na nich provedeny experimenty. Jelikož pro trénování bude použit nástroj Weka z oddílu 1.2.2, budou datasety ve formátu souboru ARFF. Pro zajištění co nejlepšího vzorku testovacích dat bude v následující oddílech proveden kvalitativní a kvantitativní rozbor jednotlivých vlastností.

2.3.1 Analýza predikátů

Analýza predikátů spočívala v určení co nejkvalitnějších dat, tedy informací, které mají největší popisný význam pro doménu, kterou reprezentují. Tento problém se nazývá Dlouhý chvost (*The Long Tail*) a byl představen Chrisem Andersonem [9]. V publikaci pojednává o tom, že se ekonomika proměňuje v závislosti na internetu tak, že přesouvá zájem z globálních více prodávaných produktů na menšinové produkty. Obrázek 2.1 znázorňuje graf Dlouhého chvostu, na kterém je zeleně označena Hlava (*Head*), která vyjadřuje 20% globálních produktů, a žlutě označený Dlouhý chvost (*Long tail*), který vyjadřuje menšinové produkty.

Podobně jako produkty v ekonomice i predikáty entit z jedné stejné domény vytvoří graf Dlouhého chvostu tak, že na vodorovné ose budou jednotlivé predikáty a na svislé ose bude procentuální zastoupení predikátů v doméně. Takto vyjádřený graf znázorňuje obecné predikáty (Hlava), které s největší pravděpodobností budou obsaženy i v entitách ostatních domén, ale také specifické predikáty (Dlouhý chvost), které vetšinou detailně popisují jednu entitu, ale pro doménu nemají význam.

K získání dat vhodných pro učení z predikátů je proto nutné určit střední oblast významu, tedy aby predikáty nebyly ani moc obecné a ani moc spe-

cifické. Pro všechny entity byl tedy proveden výpočet procentuálního zastoupení predikátů v doménách a poté vymezena oblast zájmu určením horní[40] a spodní[41] meze.

2.3.2 Analýza dct:subject

Obdobný problém jako u predikátů se vyskytl i u *dct:subject* (dále jen kategorie). Oproti predikátům nebyl problém vymezit střední oblast významu, ale pouze vymezit prostor o specifické kategorie. Po analýze všech domén a jejich kategorií bylo vyvozeno rozhodnutí, že nebude nutné nastavovat spodní mez každé doméně zvlášť, ale bude stačit určit jednu globální mez pro všechny domény. Hodnota spodní meze byla nastavena na 1.5%.

2.3.3 Analýza rdf:type

Problém *rdf:type* (dále jen typy) je oproti předchozím trochu jiný. Jelikož se jedná o třídy, které reprezentují danou entitu, nelze se jen zaměřit na procentuální zastoupení typů v doménách. Je to dáno tím, že ve většině případů se dá entita reprezentovat jen pár třídami, které se ale dají vyjádřit více ontologiemi. Proto byla veškerá pozornost zaměřena na výběr nejvhodnější ontologie pro extrakci tříd s co největším významem.

Při procházení náhodného vzorku pojmenovaných entit bylo zaregistrováno okolo pěti nejvíce se vyskytujících ontologií. Některé byly moc obecné a některé zase moc specifické, podobně jako u předchozích vlastností a problému Dlouhého chvostu. Ve střední oblasti byly ontologie, které se významově skoro shodovaly. Proto, i po doporučení vedoucího této práce, bylo usouzeno, že je vhodné vybírat typy jen z DBpedia ontologie[42].

2.4 Wikipedia a DBpedia

Jak je psáno přímo na jejich stránkách [10], Wikipedia je projekt multijazyčné, webové, volně dostupné encyklopedie, spravovaný Wikipedia Foundation a založený na modelu upravování svého obsahu samotnými uživateli. V současné době Wikipedia obsahuje více než 5 milionů článků psaných v anglickém jazyce. Tyto články nejsou tvořeny jen volným textem, ale jsou obohaceny o interní i externí odkazy spojující články mezi sebou anebo doplňující informace týkající se obsahu. Takovýto způsob vytváření a uchovávání dat je základem

[40] horní mez – Predikáty s procentuální zastoupení vyšším než horní mez nebudou brány v potaz.

[41] spodní mez – Predikáty s procentuální zastoupení nižším než dolní mez nebudou brány v potaz.

[42] DBpedia ontologie – Jmenný prostor s identifikátorem „`http://dbpedia.org/ontology/`“ a prefixem „dbo:“.

Obrázek 2.2: Náhled webové stránky DBpedia pro zdroj (článek) „Ice hockey“. Pořízeno 8. 3. 2016 jako snímek obrazovky webové stránky `http://dbpedia.org/page/Ice_hockey`.

sémantického webu, který má za cíl udržovat význam informací tak, aby byly čitelné jak pro člověka, tak i pro stroj.

Projekt založený na základě informací z Wikipedie, má na starosti crowdsourcing[43] komunita jménem DBpedia [12], jejíž cílem je extrahovat strukturovaná data z Wikipedie a nabídnout je jako volně dostupná. Tato data obsahují vlastnosti a vztahy Wikipedia článků, ale také odkazují na jiné externí datasety, které s nimi souvisejí. Kromě datových sad vytvořených na základě Wikipedie nabízí DBpedia i ontologie, vnitřní propojení s jinými sadami a online přístup ve formě *SPARQL endpointu*[44], díky kterému se lze nad daty přímo dotazovat. Náhled DBpedia stránky popisující zdroj (článek) „Ice hockey“ je vidět na obrázku 2.2.

Hlavní technickou komponentou DBpedie je extrakční framework[45], který má za úkol z textu Wikipedie extrahovat strukturovaná data a vytvořit z nich obohacené znalosti. Wikipedia obsahuje hodně volného textu, ale také strukturovaných dat ve formě infoboxů[46], kategorií, interních a externích odkazů mezi články, referencí na jazykové mutace nebo obrázků a zeměpisných souřadnic. Právě na tato data se zaměřuje DBpedia. Další důležitou součástí DBpedie je její ontologie a vlastnosti, které jsou spravované a rozšiřované komunitou uživatelů.

DBpedia se dělí do 125 jazykových verzí a dohromady popisuje přes 39

[43] Crowdsourcing – Definice ze serveru dictionary.cambridge.org „the act of giving tasks to a large group of people or to the general public, for example, by asking for help on the internet, rather than having tasks done within a company by employees“ [11].

[44] *SPARQL endpoint* – Veřejný koncový bod k připojení k datasetu.

[45] Framework – Aplikační struktura sloužící k podpoře při vývoji softwarových projektů.

[46] Infobox – Panel vyskytující se v textu článku, popisující entitu stručnými fakty.

Tabulka 2.1: Analýza domén, datasetů a vlastností entit (ukázka).

Doména	Počet entit	Horní mez	Spodní mez	Počet predikátů	Předpokládaná TaaS doména
Royalty and nobility	8	0.15	0.0	34	
Mathematics	9	0.34	0.1	22	2100 Natural sciences
Transport	175	0.47	0.1	58	0100 Politics and Administration
Food and drink	14	0.29	0.1	44	1000 Agriculture and foodstuff
Sport and recreation	208	0.54	0.15	29	2005 Social sciences
Vysvětlivky: Horní a spodní mez – Souvisí s predikáty z oddílu 2.3.1.					

milionů[47] objektů. Ze všech těchto dat a informací vzniká soubor datasetů o velikosti okolo 3 miliard RDF trojic a z toho 580 milionů jich bylo extrahováno z anglické Wikipedie.

2.4.1 Shrnutí analýzy dat

V rámci analyzování domén, datasetů a vlastností entit byla vytvořena tabulka, která dále posloužila při vývoji aplikace a vytváření experimentů nad těmito daty. Příklad analyzovaných dat lze vidět v tabulce 2.1, všechna data jsou pak k dispozici v příloze C. Další detailní analýzy týkající se dat lze nalézt na přiloženém CD nebo ve veřejném repozitáři z podkapitoly 3.4.

2.5 Analýza algoritmů klasifikace

V předchozí podkapitole 2.3 byla zvolena trénovací data a formát, v jakém budou reprezentovány. Dalším krokem je analýza klasifikačních algoritmů. Součástí této analýzy bude proveden výběr algoritmů, které budou použity v dalších částech této práce.

Při výběru bylo bráno v úvahu pět skupin klasifikátorů uvedených v oddíle 1.2.3, ze kterých bude použit, kromě perceptronových sítí, vždy jeden klasifikátor. Poté budou v experimentální části této práce klasifikátory porovnány. Výběr klasifikátorů závisel na doporučení vedoucího této práce a na článku Edwina Chena *Choosing a Machine Learning Classifier* [13] a publikaci S. B. Konsiantise *Supervised Machine Learning: A Review of Classification Techniques* [14].

[47] 39 milionů objektů – K 17. 4. 2016. Dostupné online: `https://meta.wikimedia.org/wiki/List_of_Wikipedias#Grand_Total` (17. 4. 2016).

2.5.1 Logické algoritmy

Skupinu logických algoritmů lze dále rozdělit na metody rozhodovacích stromů a klasifikátory na základě pravidel (*rule-based*).

Rozhodovací stromy (*decision trees*) klasifikují instance trénovacích dat na základě seřazení podle jejich vlastností. Instancemi dat je myšlen jeden řádek, v případě této práce datové části souboru ARFF z oddílu 1.2.2. Za vlastnosti pak považujeme atributy, například predikáty, představené v sekci 1.1.1. Každý rozhodovací strom se skládá z uzlů tvořených z vlastností jednotlivých instancí a hran skládajících se z hodnot, kterých může uzel nabývat. Kořen stromu tvoří vlastnost, která nejlépe rozděluje trénovací data. K nalezení kořene se dají využívat různé algoritmy. Sestavení rozhodovacího stromu je NP-úplný problém, a proto jsou také zkoumány různé efektivní heuristiky k sestrojení téměř optimálních rozhodovacích stromů.

Každý rozhodovací strom lze přeložit na množinu pravidel, které popisují všechny cesty stromem od kořene k listům. Nebo lze tyto pravidla vytvořit rovnou z trénovacích dat pomocí *rule-based* algoritmů. Cílem těchto algoritmů je vytvořit co nejmenší množinu pravidel, kde každé pravidlo reprezentuje jednu třídu a je vyjádřeno v disjunktní normální formě. Rozdíl mezi rozhodovacími stromy a učením na základě pravidel je, že strom vyhodnocuje průměrnou kvalitu disjunktních cest (jedna pro každou hodnotu testované vlastnosti), zatím co učení podle pravidel vyhodnocuje kvalitu sady instancí, které spadají pod vyhovující pravidlo.

Algoritmem vybraným pro účely této práce ze skupiny logických algoritmů je algoritmus Random Forest. Jedná se o algoritmus, který generuje mnoho klasifikátorů a poté seskupuje jejich výsledky. Pro tento proces využívá metody *bagging*, ve které nezáleží na pořadí konstrukce klasifikátorů a výsledek je nejčastější hodnota (modus) tříd vrácených klasifikátory. Zdrojem popisu algoritmu byla publikace *Classification and Regression by randomForest* [15].

Z názvu se dá vyvodit, že jako klasifikátory používá rozhodovací stromy a vybírá náhodný vzorek dat, ze kterých pak jednotlivé stromy konstruuje. Rozhodovací stromy mají velkou tendenci k přeučení trénovacího datasetu, to je dáno jejich nízkou odchylkou, ale naopak vysokým rozptylem. Algoritmus Random Forest redukuje rozptyl na úkor odchylky tím, že vybírá modus více stromů natrénovaných na různých částech trénovacích dat, a tak zvýší výkon výsledného natrénovaného modelu. Celý proces klasifikace probíhá ve třech krocích:

1. Náhodně vytvoří vzorky dat z původních dat.
2. Provede klasifikaci stromů z vytvořených vzorků.
3. Seskupí predikce všech stromu a vrátí jejich modus.

Ke spuštění procesu klasifikace stačí zvolit dva parametry: počet náhodných proměnných pro konstrukci stromů a počet stromů. Pro nastavení nej-

lepší kombinace parametrů lze použít evaluační metodu *cross-validation* z oddílu 1.2.4 nebo sledovat odhad chybovosti. Hodnota odhadu chybovosti je střední hodnota chyby predikce trénovacích vlastností v_n, s použitím pouze těch stromů, které byly vytvořeny ze vzorku dat. Tento vzorek však neobsahuje vlastnosti v_n. Tato chyba se také nazývá *out-of-bag* a z pohledu nastavení parametrů sledujeme přirozeně čím menší, tím lepší.

2.5.2 Perceptronové sítě

Algoritmy perceptronových sítí se dělí do dvou skupin na jednovrstvé a vícevrstvé.

Jednovrstvé perceptronové sítě se nejčastěji používají pro učení z dávek trénovacích instancí a algoritmus učení probíhá opakovaně nad trénovacími daty do té doby, než je nalezen vektor predikce, který vyhovuje všem trénovacím instancím. Tento vektor je pak použit k predikci třídy, v případě této práce domény, testované instance. Definice těchto sítí podle S. B. Kotsiantise: „Pro vstupní hodnoty vlastností x_1 až x_n a jejich váhy/predikce w_1 až w_n (typicky reálná čísla v intervalu $[-1, 1]$), perceptron vypočítá váženou sumu $\sum_i x_i w_i$ a jako výstup vrátí 1, pokud je suma větší než nastavený práh, anebo 0, pokud je menší.“[48]

Vícevrstvé perceptronové sítě, známé také jako umělé neuronové sítě, řeší omezení perceptronů, pro než platí, že jsou schopny najít řešení jen pro lineárně oddělitelné množiny instancí. Tyto sítě obsahují mnoho jednoduchých jednotek (perceptronů), které jsou mezi sebou propojeny. Jednotky se pak rozdělují do 3 skupin na vstupní, výstupní a skryté. Mechanismus trénování pak probíhá stejným způsobem jako u jednovrstvých sítí.

Žádný z algoritmů perceptronových sítí nebyl pro využití v této práci zvolen, protože nebyl vhodný při velikém množství atributů (vlastností) trénovacích instancí.

2.5.3 Statistické algoritmy

Statistické učící algoritmy jsou založené na pravděpodobnostním modelu, jehož pravděpodobnosti určují, do jaké míry instance náleží dané třídě. Mezi nejznámější statistické algoritmy patří Bayesovské sítě.

Strukturou Bayesovské sítě je orientovaný acyklický graf tvořen z pravděpodobností vztahů všech vlastností. Každý uzel grafu koresponduje s jednou vlastností. Proces učení je typicky rozdělený do dvou částí: vytvoření orien-

[48] V textu přeloženo do češtiny. Definice v původním znění: „If x_1 through x_n are input feature values and w_1 through w_n are connection weights/prediction vector (typically real numbers in the interval $[-1, 1]$), then perceptron computes the sum of weighted inputs: $\sum_i x_i w_i$ and output goes through an adjustable threshold: if the sum is above threshold, output is 1; else it is 0.“; [14, str. 254].

tovaného acyklického grafu a nastavení jeho parametrů. Tyto pravděpodobnostní parametry jsou zapsané do množin tabulek, jedna pro každou vlastnost. Také lze sestrojit Bayesovskou síť použitím statistických testů, hledáním podmíněných nezávislých vztahů mezi vlastnostmi a použitím těchto vztahů jako omezení sítě.

Pro potřeby této práce byl z této skupiny algoritmů vybrán algoritmus Naive Bayes. Je to velmi jednoduchá Bayesovská síť, která je tvořená orientovaným acyklickým grafem s jedním rodičem a nezávislými potomky. Učení tímto algoritmem spočívá ve výpočtu podmíněné pravděpodobnosti všech atributů A_i vůči třídě C ze všech instancí trénovacích dat. Proces klasifikace poté vypočte pravděpodobnost třídy C na základě atributů $T_1, \ldots, T_n$ testovací instance a vybere třídu s největší pravděpodobností (provede predikci). Tento algoritmus je popsán ve článku *Bayesian Network Classifiers* [16], ze kterého bylo čerpáno. Tato predikce je možná díky nezávislosti všech atributů A_i mezi sebou vzhledem třídě C.

Naive Bayes klasifikátor můžeme reprezentovat pomocí Bayesovské sítě tak, že bude obsahovat jeden cílový uzel, jehož potomci budou všechny ostatní vstupní uzly, které jsou nezávislé. Toto se dá vyjádřit vztahem:

$$P(C, A_1, \ldots, A_n) = P(C)P(A_1|C) \ldots P(A_n|C) \tag{2.1}$$

Rozdílem oproti jiným algoritmům (například rozhodovací stromy) je, že při klasifikaci neprohledává prostor hypotéz, ale pouze vypočítá pravděpodobnosti podle četnosti výskytů jednotlivých atributů.

Hlavní předností algoritmu Naive Bayes je výpočetní čas trénování. To je dáno tím, že algoritmus používá operace násobení a tu lze převést na sumu přes logaritmy, která je výpočetně méně náročná.

2.5.4 Instanční algoritmy

Základem instančních algoritmů je zpomalování procesů indukce a generalizace, dokud neproběhne klasifikace. To znamená, že oproti algoritmům ze skupin rozhodovacích stromů nebo perceptronových sítí spotřebují méně výpočetního času v přípravné části, ale naopak více v části klasifikační. Hlavním algoritmem instančních učení je algoritmus Nejbližšího souseda (*Nearest Neighbour*).

Principem NN, neboli *Nearest Neighbour*, je výskyt instancí datasetu co nejblíže ostatním instancím, které mají podobné vlastnosti. Instance si lze představit jako body v n-dimenzionálním prostoru, kde jsou jednotlivé dimenze tvořeny vlastnostmi instancí. Důležitou roli zde proto hraje výběr vzdálenostní funkce. Nevýhodou tohoto algoritmu je náročnost na spotřebu paměti a také závislost na podobnostní funkci.

Algoritmus k-Nearest Neighbours (dále jen kNN), který byl vybrán pro potřeby této práce, navazuje přímo na instanční klasifikátor nejbližších sousedů. Tento algoritmus rozšiřuje NN o možnost nastavení konstanty k pro rozšíření

okolí na výběr k nejpodobnějších instancí. Speciální případ nastane, když hodnotu k nastavíme na 1, a kNN tedy bude stejné jako NN. Při popisu algoritmu kNN bylo čerpáno z publikace *KNN Model-Based Approach in Classification* [17].

Jedná se o nejjednodušší algoritmus využívaný ke strojovému učení a patří do skupiny instančních algoritmů, ale také mezi takzvané *lazy learning* algoritmy. Stejně jako u NN lze u kNN používat různé metody k výpočtu vzdálenosti dvou instancí, kdy je nejčastěji používanou Euklidova vzdálenostní funkce. V procesu učení se jen vyberou všechny trénovací instance a umístí se do N-rozměrného prostoru. Dále se v klasifikaci vloží testovací prvek do stejného prostoru a provede výpočet vzdálenosti ke všem prvkům s vrácením k nejbližších sousedů. Poté se hledaná třída testovacího prvku určí podle největšího zastoupení třídy nejbližších sousedů. Z tohoto důvodu se dá říci, že je algoritmus závislý na volbě konstanty k, a tak bude nutné si dát na jeho nastavení v experimentální části pozor.

Kromě volby k a vzdálenostní funkce se dá také volit, zda při výběru nejbližších sousedů provádět vážení vzdálenostní funkce, například $\frac{1}{d}$ nebo $1 - d$, kde d je vypočtená vzdálenost. Tato metoda snižuje pravděpodobnost kolize, což je situace, kdy pro testovanou instanci nelze vybrat k nejbližších sousedů.

2.5.5 Support Vector Machines (SVM)

Jsou to techniky založené na optimálním rozdělení trénovacích dat hledáním nadroviny. Tato rozdělující nadrovina je lineární funkcí nad vlastnostmi a na její popis stačí pouze nejbližší body (*support vectors*). Rozděluje data do dvou tříd, které leží v opačných poloprostorech, a mezi nimi je co největší odstup (*margin*). Správné trénování znamená vyřešit problém n-dimenzionálního kvadratického programování, a to obnáší mnoho maticových operací a spotřebu paměti.

Tato omezení řeší Sequential Minimal Optimization (SMO), který byl i proto vybrán do skupiny klasifikačních algoritmů použitých v této práci. Tento iterativní algoritmus optimalizuje problém metody SVM. Optimalizační problém vychází z binární klasifikace, kterou SVM řeší kvadratickým programování závislým na Lagrangeových multiplikátorech. SMO problém rozděluje na menší podproblémy závislé pouze na dvou Lagrangeových multiplikátorech, jejichž výhodou je, že se dají řešit analyticky. K tomuto procesu není potřeba žádný maticový prostor navíc ani žádné numerické výpočty kvadratického programování. Algoritmus probíhá v následujících krocích:

1. Najít první Lagrangeův multiplikátor, který porušuje KKT[49] podmínky pro optimalizační problém.

[49] KKT – Karush-Kuhn-Tucker. Podrobněji v publikaci *Constraint Quailifications for Nonlinear Programming* [18].

2. Najít druhý Lagrangeův multiplikátor a optimalizovat ho s prvním.

3. Opakovat první dva kroky, dokud nenastane konvergence.

K nalezení Lagrangeových multiplikátorů se používají dvě heuristiky (každá pro jeden).

Na rozdíl od ostatních klasifikátorů je v případě počtu tříd větší než dva pro testovanou instanci dat určena její pravděpodobnost predikce spárováním podle metody *pairwise coupling*[50], což je dáno problémem binární klasifikace. Více se tímto problémem zabývá John C. Platt ve své práci *Sequential Minimal Optimization: A Fast Algorithm for Training Support Vector Machines* [20], která byla hlavním zdrojem tohoto popisu.

2.6 REST služba

Hlavním výstupem této práce by mělo být automatické učení domén entit formou služeb REST, proto bude v této sekci představeno, co to REST služby jsou a jaké výhody má jejich používání v oblasti reprezentace výstupu aplikací. Jako zdroj popisu REST služeb byly použity přednášky docenta Tomáše Vitvara [21].

Representational state transfer (REST), navržený a publikovaný v roce 2000 Royem Fieldingem, je architektonický styl pro distribuované interaktivní (*hypermedia*) systémy. Tento styl je zaměřený na přístup ke zdrojům a definuje omezení, která pomáhají vytvářet dobře navržené aplikace. Omezení nebo lépe principy, jsou komunikace klient/server, bezestavovost, využití cache[51], dělení systému do vrstev a jednotné rozhraní. O tom podrobněji pojednává 5. kapitola *Representational State Transfer (REST)* z publikace [22]. Nejvíce nás bude zajímat to, že REST je zaměřen na zdroje. Zdrojem mohou být data, ale i stavy aplikace. Každý zdroj má svůj identifikátor. Jako zdroj s identifikátorem si můžeme představit DBpedia entitu ve formátu RDF trojic z předchozího oddílu1.1.1. K těmto zdrojům REST používá přístup pomocí základních metod CRUD (*Create*, *Retrieve*, *Update*, *Delete*), které implementuje pomocí protokolu HTTP. Při komunikaci si klient se serverem posílají zprávy ve formě metadat a dat. Názornou ukázkou může být REST služba provozovaná serverem DBpedia, kdy klient odešle HTTP GET na URL[52] zdroje, například entity *Berlin*, s hlavičkou obsahující formát odpovědi (metadata). Server požadavek zpracuje a popis entity (data) v daném formátu odešle zpět klientovi.

V této práci by měla REST služba komunikovat obdobně jako služba běžící na serveru DBpedia. Klient pošle požadavek s identifikátorem pojmenované

[50] *pairwise coupling* – Podrobněji v publikaci *Pairwise Coupling and Pairwise with Majority Voting* [19].

[51] Cache – Vyrovnávací paměť.

[52] URL – Uniform Resource Locator – Specifický typ URI, který definuje umístění webového zdroje v počítačové síti.

entity a formátem odpovědi, server požadavek přijme. Nejdříve však z modelů predikuje naučenou doménu a pak teprve odešle výsledná data zpět klientovi. Z tohoto pohledu je nutné určit, jak bude REST služba spojená s touto prací přijímat a odesílat požadavky.

Prvním krokem je určit, jakým způsobem bude klient identifikovat zdroj s tím, že identifikátor entity bude zachován stejný podle DBpedie. V tomto případě je identifikátorem myšlena část v URL reprezentující název entity. Například pro entitu *Ice hockey*, která je reprezentována pod URL[53], je to řetězec *Ice_hockey*. Klient může buď odeslat požadavek, kdy identita entity bude součástí dat, nebo bude požadavek odeslán na adresu, jejíž součástí bude identita entity. Z důvodu dodržení principů REST a také zachování uživateli zažité identifikace ze serveru DBpedia byla zvolena druhá možnost.

Krokem druhým je zvolit formu odpovědi. Z kontextu obsahu by odpověď měla určitě obsahovat DBpedia identifikátor pojmenované entity, dále pak predikovaný název domény a procentuální ohodnocení, na kolik doména entitě vyhovuje. Tato procenta jsou součástí vyhodnocení predikce knihovny Weka. Dále je pak důležité, jaké formáty odpovědí bude REST služba nabízet. Z pohledu uživatele není až tak důležité řešit vzhled, tudíž byl zvolen volný text. Pro možnost využití dalšími aplikacemi byl zvolen formát JSON, který je doporučovaným formátem pro webové aplikace a REST služby.

[53]Identifikátor entity *Ice hockey*: „`http://dbpedia.org/resource/Ice_hockey`“.

Kapitola 3

Návrh a implementace

V předchozích kapitolách byl představen a analyzován problém, kterým se tato práce zabývá. V této kapitole bude popsán postup při návrhu a implementaci komponent výsledných aplikací a také zde budou představeny technologie, které byly použity a návod, jak postupovat při instalaci a spuštění obou aplikací.

V odstavci výše je možné zaznamenat, že pojednává o dvou aplikacích. Je to dáno tím, že bude celý problém rozdělen na dvě části (dvě aplikace). První aplikace slouží hlavně k vytvoření trénovacích datasetů pro import do programu Weka z oddílu 1.2.2, dále bude nazývána Tool. Druhá aplikace je implementací REST služby k predikci domén entit pomocí natrénovaných modelů, ta bude nazývána REST API.

3.1 Použité technologie

V úvodu této části je nutné popsat, jaká technologie při implementaci bude využívána a proč. Jako programovací jazyk byl zvolen jazyk Java[54]. Důvodem je použití Java knihoven nástroje Weka. Tyto knihovny nabízí stejnou funkcionalitu jako grafické rozhraní, ale i využití všech tříd a metod k práci s modely a datasety. Po doporučení vedoucího práce bylo rozhodnuto, že budou obě aplikace založené na nástroji *Apache Maven*[55], který slouží ke správě Java projektů. Dále byly v této práci použity knihovny *RDF/HDT*[56] pro práci s datasetem DBpedia, *Apache Jena*[57] ke zpracování sémantického webu, *JAX-RS*[58] k vytvoření REST služby a další menší podpůrné nástroje. Tyto nástroje a knihovny budou více představeny v následujících oddílech, kde budou popsána konkrétní využití v mých aplikacích.

[54]Více informací online: `https://www.java.com/` (25. 3. 2016).
[55]Více informací online: `https://maven.apache.org/` (25. 3. 2016).
[56]Více informací online: `http://www.rdfhdt.org/` (25. 3. 2016).
[57]Více informací online: `https://jena.apache.org/` (25. 3. 2016).
[58]Více informací online: `https://jax-rs-spec.java.net/` (25. 3. 2016).

3.1.1 Java

Java je programovací jazyk, který je objektově orientovaný a vychází převážně z jazyka *C++*. Byl vytvořený Jamesem Goslingem v roce 1995 ve firmě Sun Microsystems (dnes patří Oracle Corporation). Program napsaný v jazyce Java je interpretovaný pomocí virtuálního počítače (JVM)[59], proto je nezávislý na konkrétním operačním systému nebo hardwaru. Oproti jazyku *C++* je čistě objektově orientovaný a snaží se o co největší jednoduchost, například nepoužíváním ukazatelů nebo odstranění vícenásobné dědičnosti. Dále není překládán přímo do nativního kódu, ale do mezikódu (*bytecode*), který je vykonáván virtuálním strojem, proto se řadí mezi jazyky „write once, run anywhere“[60], což znamená, že je multiplatformní bez nutnosti rekompilování kódu. Více o historii tohoto jazyka lze nalézt v publikaci *Historie a vývoj jazyka Java* [23], ze kterého byly čerpány informace pro tento text, anebo přímo na stránkách společnosti Oracle[61].

3.1.2 Apache Maven

Apache Maven je nástroj sloužící ke správě a vytváření projektů založených na jazyce Java. Cílem tohoto projektu je usnadnění a zpřehlednění práce s Java projekty z pohledu často opakovaných procesů od procesu sestavení (*build*), přes správu závislostí, testování a dokumentace, až po spouštění aplikace, migraci nových verzí a správu pluginů. Všechny tyto informace o projektu uživatel nastavuje do souboru POM (*project object model*), který je programem *Maven* zpracován a na jeho základě jsou provedeny všechny potřebné úkony. Pokud uživatel pracuje s více *Maven* projekty, je tento nástroj schopný efektivně pracovat s jejich závislostmi a pluginy tak, aby ušetřil co nejvíce paměti. Hlavní výhodou je přenositelnost projektů mezi operačními systémy i vývojovými prostředími.

Díky výše popsaným vlastnostem byl nástroj použít i pro správu aplikací této práce. Dalším argumentem pro zvolení tohoto nástroje bylo doporučení vedoucího práce z důvodu možného zveřejnění výsledného projektu ve veřejném repositáři, o tom v kapitole 3.4. Více informací lze nalézt přímo na stránkách projektu *Apache Maven*[62].

3.1.3 RDF/HDT

Použití tohoto nástroje úzce souvisí s výběrem zdroje strukturovaných dat DBpedie. Předtím, než bude nástroj představen a popsán, budou vysvětleny možnosti přístupu k datasetu DBpedie a řešení, které bylo zvoleno. Varianty byly dvě. První z nich byla možnost vzdáleného připojení k DBpedia *SPARQL*

[59] JVM – Java Virtual Machine – Virtuální stroj, který zpracovává a spouští Java mezikód.
[60] „write once, run anywhere“ – česky: „napiš jednou, spusť kdekoliv“.
[61] Dostupné online: `https://www.java.com/` (25. 3. 2016).
[62] Dostupné online: `https://maven.apache.org/what-is-maven.html` (25. 3. 2016).

endpoint a mít tak k dispozici všechna nabízená data, ovšem pouze za předpokladu připojení k internetu. Druhá z možností bylo uložení datasetů na disk a připojování se k nim lokálně. Byla zvolena druhá možnost i za cenu obětování velkého množství místa na disku. A tímto se dostáváme k *RDF/HDT*.

HDT (*Header*, *Dictionary*, *Triples*) je otevřený binární serializační formát sloužící ke kompresi RDF datasetů. Zmenšuje jejich velikost a tak usnadňuje jejich ukládání a sdílení na webu. *RDF/HDT* nabízí knihovny pro přístup k RDF trojicím, přes které se pak další knihovny nad daty dotazují nebo s nimi manipulují (viz následující oddíl 3.1.4). Důležité je ale řešit výhody a nevýhody oproti přístupu k online *SPARQL endpointu*. Protože je HDT indexovaný, rychlosti odpovědí na dotazy přes *SPARQL endpoint* nebo HDT soubor jsou srovnatelné. Nevýhodou by mohla být synchronizace aktuálních dat nabízených přes *SPARQL endpoint* oproti souboru HDT. Vzhledem k tomu, že proces převodu Wikipedia dat na DBpedia datasety probíhá zhruba dvakrát ročně, není to až takový problém.

Z pohledu celkové velikosti, kterou zabírají HDT soubory (dataset plus indexy) na disku, se jedná o 10.2 GB, což se, s přihlédnutím na výše popsané výhody, dá považovat za přijatelné. Další informace o formátu HDT a nástrojích k jeho zpracování jsou k dispozici na webu projektu *RDF/HDT*[63].

3.1.4 Apache Jena

Jedná se o framework postavený na jazyce Java, který slouží k vytváření a zpracování aplikací sémantického webu. Pro účely této práce budou základní knihovny tohoto frameworku využity k napojení se na HDT dataset DBpedie s následným dotazováním a zpracováním výsledků. Nástrojů tohoto typu existuje více, ale byl vybrán tento, protože má přímou podporu právě pro kompresní formát HDT. Více informací lze nalézt na stránkách projektu[64].

3.1.5 JAX-RS

Projekt *Java API for RESTful Services* (*JAX-RS*) slouží k vytváření webových služeb v jazyce Java s podporou REST. Ke specifikaci služeb (cesty, metody, hlavičky, atd.) používá v kódu anotace. U výběru těchto knihoven z velké části také rozhodla vysoká oblíbenost v komunitě vývojářů aplikací v jazyce Java a také dobrá dokumentace a podpora od společnosti Oracle. Specifikaci programového rozhraní lze nalézt na portálu Java[65].

Z použití knihoven *JAX-RS* vyplývá, že výsledná aplikace poskytující REST služby bude webová aplikace, tudíž bude potřebovat ke svému běhu webový server. Byl zvolen server *GlassFish*[66] od společnosti Oracle, a to z toho

[63]Dostupné online: `http://www.rdfhdt.org/what-is-hdt/` (25. 3. 2016).
[64]Dostupné online: `https://jena.apache.org/about_jena/about.html` (25. 3. 2016).
[65]Dostupné online: `https://jax-rs-spec.java.net/` (25. 3. 2016).
[66]Více informací online: `https://glassfish.java.net/` (25. 3. 2016).

důvodu, že při psaní Java kódu obou aplikací bylo použito vývojové prostředí *NetBeans*[67], jehož je server součástí. Pokud ovšem uživatel nepoužívá toto prostředí, je nucen instalovat server dodatečně, proto byla vybrána i alternativa, kterou je *Embedded GlassFish*[68], jenž se instalovat nemusí a lze spustit jako knihovna na JVM. *Embedded GlassFish* má oproti serveru *GlassFish* určitá omezení, ale pro rychlé spuštění aplikace této práce postačí. Výhodou *Embedded GlassFish* je i integrace s nástrojem *Maven*, tudíž lze jednoduše jedním příkazem spustit server s REST aplikací najednou.

3.1.6 Další podpůrné nástroje

Podpůrnými nástroji jsou myšleny knihovny, které zpracovávají především data. Tyto nástroje budou shrnuty v jednom oddíle, protože jejich použití zásadně neovlivňuje aplikace této práce a patří mezi nástroje, které jsou programátory velice využívané a dá se do jisté míry říci, že i známé. Jedná se o nástroje *jsoup*[69] a *JSON.simple*[70].

Knihovna *jsoup* slouží k parsování[71] HTML stránek. V kontextu této práce bude knihovna použita ve spojení s aplikací Tool, kde pomocí ní jsou stahovány URL identifikátory stránek (entit) pro jednotlivé domény z Wikipedia Featured Articles.

Nástroj *JSON.simple* je použit v obou aplikacích, a to při zakódování a dekódování textu formátu JSON. V REST API je knihovna používána při zpracování výsledku pro klienta, který požadoval odpověď ve formátu JSON. V Tool je tato knihovna použita pro uchování analytických dat anebo k vytvoření zálohy načtených informací o doménách a entitách (viz oddíl 3.2.8).

3.2 Tvorba trénovacích datasetů

Návrh a implementace této práce začala aplikací Tool (dále jen Tool) z důvodu vyplývajícího z předchozího textu, především tedy kvůli vytvoření trénovacích datasetů pro další zpracování.

3.2.1 Pochopení problému a první vytvoření modelu

Pro lepší pochopení problematiky zpracování dat byly zvoleny základní domény z prostředí, ve kterém se pohybuji nebo jsou součástí každodenního života. Z nich budou vytvořeny trénovací datasety. Pro srovnání s Featured Articles byly zvoleny domény *Sport and recreation*, *Music* a *Food and drink*.

[67] Více informací online: `https://netbeans.org/` (25. 3. 2016).
[68] Více informací online: `https://embedded-glassfish.java.net/` (25. 3. 2016).
[69] Více informací online: `http://jsoup.org/` (25. 3. 2016).
[70] Více informací online: `https://code.google.com/archive/p/json-simple/` (25. 3. 2016).
[71] Parsování HTML – Syntaktická analýza HTML dokumentu.

Pro každou doménu byl vytvořen seznam 20 entit, které jsem hledal procházením stránek Wikipedie a u kterých jsem se snažil udržet nezávislost oproti entitám z Featured Articles.

Pro základní vytvoření trénovacího datasetu jsou potřeba 4 kroky:

- Připojení se ke zdroji dat.
- Vytvoření dotazu.
- Zpracování dat.
- Výpis získaných informací ve formátu ARFF.

Tyto kroky budou více přiblíženy v následujících odstavcích.

Jak už bylo popsáno v oddíle 3.1.3, za zdrojová data byl zvolen dataset DBpedia v komprimovaném formátu HDT. K těmto datům se přistupuje pomocí knihovny *Jena*, která dataset načte jako model z grafu, který vrací knihovna *RDF/HDT*. S tímto modelem lze následně provádět všechny operace, které definuje RDF. Pro potřeby této práce postačí pouze vytvoření vhodného dotazu k extrakci informací o jednotlivých entitách.

K provádění dotazů nad *Jena* modelem slouží knihovna *Jena ARQ*, která je součástí balíčku knihoven *Apache Jena*. *Jena ARQ* zpracovává SPARQL dotazy a jako výsledek vrací seznam RDF uzlů, které obsahují požadované informace. Tento proces je nutné provádět v cyklu, tudíž pro každou entitu zvlášť získáme seznam informací, nad kterými budeme provádět trénování. Z toho důvodu jsem se rozhodl vytvořit třídu *Entity*, která v sobě bude uchovávat URL identifikátor entity a seznamy informací získaných z dotazů.

V podkapitole 2.3 jsou popsány informace, nad kterými bude prováděno trénování. V počáteční fázi budou zpracovány nejprve predikáty. Následně po vytvoření a porovnání datasetů ručně nalezených entit a entit Featured Articles se bude pokračovat v získávání dalších informací.

Po těchto krocích nastavení byl vygenerován první soubor s trénovacími instancemi ve formátu ARFF, jehož správnost byla otestována v programu Weka. Syntakticky byl dataset s trénovacími daty validní, ovšem při procházení atributů instancí byly zpozorovány příliš obecné vlastnosti, tedy problém Dlouhého chvostu z oddílu 2.3.1. Z toho bylo usouzeno, že je potřeba nejprve informace o entitách nějakým způsobem zanalyzovat a poté zavést pravidla pro extrakci informací s největším významem.

Jak už bylo popsáno v závěru oddílu 2.3.1, muselo být provedeno porovnání procentuálního zastoupení predikátů ve všech entitách domén. Tedy vyřešit problém uložení všech predikátů s informací o počtu výskytů nebo procentuálním zastoupení v doméně. Z tohoto důvodu jsem vytvořil třídu *Domain*, jejímž hlavním úkolem je udržení základních informací o doméně (jméno, entity), ale i o analytických informacích (predikáty s počtem výskytů v doméně). Pro účel práce s predikáty obsahuje třída *Domain* pomocnou strukturu *MyInfo*, kterou

tvoří dvě proměnné: *name* a *percent*. V předchozích větách je počítáno pouze s predikáty, jelikož se týkají tohoto oddílu, ovšem z názvu pomocné struktury lze usoudit, že třída *Domain* bude uchovávat analytické informace i o zbylých dvou vlastnostech (*dct:subject*, *rdf:type*).

Pomocí výše popsaných struktur byly uloženy analyzované informace, na kterých záviselo vyřešení problému Dlouhého chvostu. Jelikož jsem nenašel způsob, jak automaticky „ořezat“ množinu predikátů, byl pro každou doménu pořízen výpis predikátů sestupně podle procentuálního výskytu a z něj byly určeny horní a spodní meze výseče, která bude dále použita pro trénování. Výběr mezí byl zvolen podle vlastního uvážení. Toto bylo možné si dovolit, protože identifikátory predikátů splňují myšlenku sémantického webu z oddílu 1.1.1 a jsou tedy „samopopisné“. Po této analýze jsem se rozhodl, že součástí třídy *Domain* a jejího konstruktoru budou i proměnné *predicateUpperBound* a *predicateLowerBound*, které budou při exportu trénovacích instancí do souboru ARFF vymezovat predikáty s největším významem.

3.2.2 Rozšíření o další vlastnosti

Stejným způsobem jako v předchozím oddíle byly třídy *Entity* a *Domain* rozšířeny o zbylé vlastnosti *dct:subject* a *rdf:type*. Rozdíl byl pouze ve výběru výseče informací s největším významem, který byl popsán v oddílech analýzy 2.3.2, respektive 2.3.3. Oproti predikátům nebyly použity proměnné pro horní a spodní meze, protože volba výseče byla jednodušší. Zbylé kroky se nezměnily.

Po doplnění kódu o tyto vlastnosti byly k dispozici 3 datasety určené pro trénovaní modelů, které už bylo možné mezi sebou porovnávat. Další fází vývoje aplikace byl návrh a implementace nástroje pro zpracování informací z Featured Articles.

Co nesmí být v tomto oddíle opomenuto, je detailnější popis dotazování se nad datasetem DBpedie. Výše už bylo nastíněno, že dotazy jsou formulovány a prováděny knihovnou *Jena ARQ*. Nebyly však neuvedeny informace ohledně SPARQL dotazu, kterým jsou data získávána. Dotazy pro všechny vlastnosti jsou v zásadě jednoduché, jde pouze o to získat seznam unikátních hodnot o aktuální entitě[72]. Proto v projekční části dotazu figuruje slovo *DISTINCT*. Dále je dotaz cílen na informaci, která má být získána (*?predicate* pro predikáty, *?object* a *?subject* pro *dct:subject* a *rdf:type*). Důležité je zmínit, že je nutné pro každou entitu provádět dotazy dva, jelikož chceme získat informace o entitě „z obou stran“, to znamená, že pokud budeme chtít získat pro entitu vlastnosti *rdf:type*, budou nás zajímat objekty, pro které je entita subjektem, ale i subjekty, které mají vybranou entitu jako objekt. Příklady obou SPARQL dotazů v Java kódu pro *rdf:type* vypadají takto:

```
String queryString1 =
```

[72] Aktuální entitou je myšlena právě vybraná entita ze seznamu, který je procházen cyklem. Tzn. jednotlivý dotaz pro jednotlivou entitu.

```
  "PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>" +
  "SELECT DISTINCT ?object " +
  "where { " +
  "<"+entity.getUrl()+"> rdf:type ?object . " +
  "}";
String queryString2 =
  "PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>" +
  "SELECT DISTINCT ?subject " +
  "where { " +
  "?subject rdf:type <"+entity.getUrl()+"> . " +
  "}";
```

Proměnné *queryString1* a *queryString2* jsou dále použity knihovnou *Jena ARQ* a výraz *entity.getUrl()* vypíše identifikátor entity, pro kterou mají být dotazy provedeny.

3.2.3 Seznámení se strukturou Featured Articles

Postup získávání informací o entitách Featured Articles byl stejný jako v předchozím případě u mnou nalezených entit. Rozdíl byl pouze ve vytváření seznamu identifikátorů entit, který byl potřeba extrahovat z Wikipedia stránky.

Z úvodu této kapitoly víme, že jsem se pro začátek snažil porovnat trénovací data tří domén s entitami z Featured Articles a s entitami mnou nalezenými. Pro získání identifikátorů entit byl použit nástroj *jsoup* z oddílu 3.1.6, ovšem k přesnému získání seznamu identifikátorů bylo potřeba nejprve analyzovat DOM[73] Wikipedia stránky, poté zjistit informace o identifikaci domén a nakonec určit dotaz, který nástroj *jsoup* použije k extrakci identifikátorů.

Po analýze Wikipedia stránky byly nalezeny identifikátory domén v elementech *span*, které byly vždy potomky elementů *h2*. Následně se po výskytu této dvojice elementů objevil seznam odkazů na jednotlivé stránky (entity) Featured Articles pro danou doménu. Z těchto informací byl zkonstruován dotaz, který byl ve formě textu předložen nástroji *jsoup* k selekci. Provedení selekce nad dotazem vypadá takto:

```
doc.select("h2:has(span[id="+anchor+"]) + p a");
```

Proměnná *doc* reprezentuje Wikipedia dokument a proměnná *anchor* je identifikátor domény. Následující proces probíhal stejně jako u entit mnou nalezených. Takto byly získány další tři trénovací datasety, které jsem mezi sebou porovnal a získal tak představu o procesu klasifikace a evaluace. Porovnání těchto datasetů bude dále probíráno v experimentech v kapitole 4.

V této fázi návrhu a implementace byla aplikace připravena vytvářet datasety trénovacích instancí pro všechny vlastnosti entit (predikáty, *dct:subject*

[73] DOM – Document Object Model – Objektově orientovaná reprezentace dokumentu HTML.

a *rdf:type*), ale jen pokud byly zadány identifikátory domén přímo v kódu. Proto dalším krokem bylo vytvoření automatického stahování všech domén z Featured Articles s následným vytvořením trénovacích datasetů.

3.2.4 Získání domén Featured Articles

Počátek návrhu této funkcionality vycházel z faktu, že celý proces musí být automatický a nejlépe bez žádného zásahu uživatele aplikace. Ke stažení identifikátorů domén byl opět použit nástroj *jsoup*, ale tentokrát se nemusela procházet celá stránka, ale stačilo pouze projít horní oblast s obsahem. Dotaz pro selekci vypadá takto:

```
doc.select("#mw-content-text > table > tbody > tr:eq(1) p > a");
```

Po načtení seznamu domén by následoval proces popsaný výše, tedy získání informací pro všechny entity, analýza procentuálního zastoupení predikátů a nakonec výpis datasetů. Ovšem zde se vyskytl problém při určení horních a spodních mezí pro selekci predikátů k trénování pouze nad informacemi s největším významem. Protože analýzu predikátů a určení mezí v aplikaci provádí uživatel, nelze celý proces provést automaticky. Musí být použit minimálně jeden mezikrok, který určí výseč významných predikátů.

Z tohoto důvodu jsem se rozhodl přidat do aplikace pomocnou třídu *getFAIds*, která se postará o stažení identifikátorů domén. Na základě této třídy byly vytvořeny instance jednotlivých domén, pro které jsem získal všechny entity a analyzoval jejich predikáty. Poté byla všechna analyzovaná data uložena do souborů, aby mohla být určena horní a spodní mez.

Po určení mezí bylo provedeno rozšíření předchozího kódu o definice zbylých domén a byly vygenerovány trénovací datasety pro všechny domény a vlastnosti entit Featured Articles.

Předchozí text o návrhu a implementaci byl věnován hlavně získání domén a entit a vytvoření trénovacích datasetů. V následujících oddílech bude více popsána struktura aplikace, použití návrhových vzorů, konfigurační parametry, zálohování a rozšíření domén s malým počtem entit.

3.2.5 Použití návrhového vzoru Strategy

S rostoucí složitostí kódu bylo nutné se zamyslet nad dodržením principů vývoje[74] v objektově orientovaném jazyce, a to hlavně kvůli jednoduchému rozšíření funkcionalit a zachování principu *single responsibility*. V tomto ohledu jsem se zaměřil na podobné části kódu, které ale ve výsledku konstruují stejné struktury. Tyto části se dají rozdělit do tří skupin, z toho dvě na sebe navazují.

V první řadě jsem se zaměřil na problém různých zdrojů dat, konkrétně na zdroje doménových entit (mnou nalezené a Featured Articles entity). Přesněji

[74] *The Principles of OOD* – Dostupné online: `http://butunclebob.com/ArticleS.UncleBob.PrinciplesOfOod/` (7. 4. 2016).

bylo nutné vytvořit nějaký objekt, který bude obsahovat všechny domény (instance třídy *Domain*) tak, aby se mohl vygenerovat trénovací dataset formátu ARFF. Pokud by byla tato problematika řešena pro každý zdroj dat zvlášť, prováděly by objekty stejné akce, tudíž by se v aplikaci vyskytoval ten samý kód dvakrát. Proto byla vytvořena abstraktní třída *DomainContainer*, která se stará o společné metody a která je rodičem tříd *MyDomainContainer* a *FADomainContainer*. Tyto potomci třídy *DomainContainer* obsahují pouze metody k získání základních informací o doménách a entitách. Takto navržená struktura tříd umožňuje jednoduché přidání nového (nebo upraveného) zdroje dat doménových entit.

Další skupinou podobných konstrukcí jsou akce zpracovávající vlastnosti entit (predikáty, *rdf:type* a *dct:subject*). Tento problém by mohl být vyřešen „naivně“, tedy třída *Entity* by obsahovala metody k získání informací přímo ve svém těle a pokud by nastala situace, že bychom potřebovali provést úpravu nebo přidat novou vlastnost, museli bychom kód upravovat na více místech (porušení principu *single responsibility*). Stejná situace nastává u analýzy vlastností, popsané v podkapitole 2.3 a oddíle 3.2.1, kde se ovšem problém vztahuje ke třídě *Domain*. V obou případech jsem se rozhodl použít upravený návrhový vzor *Strategy*. V následujících odstavcích bude popsána problematika nejprve pro třídu *Entity* a poté pro třídu *Domain*.

Aby byla oddělena závislost třídy *Entity* na vlastnostech entit, byl vytvořen interface *EntityInformationStrategy*, jehož jednotlivé implementace slouží k získání dat vždy pro jednu vlastnost, například *EntityPredicateStrategy* stahuje pouze seznam predikátů dané entity.

Stejně jako u třídy *Entity* byla zpracována třída *Domain*. Rozdíl byl pouze v tom, že interface *InformationStrategy* má na starosti zanalyzované vlastnosti, z kterých se poté použijí pouze ty s největším významem. Příkladem konkrétní implementace tohoto interface je třída *PredicatesStrategy*, která provádí analýzu predikátů.

3.2.6 Struktura aplikace

V návaznosti na předchozí oddíl je nutné blíže popsat aplikaci Tool z hlediska celkové struktury, vstupních argumentů, instalace a spuštění. Návrh aplikace byl od začátku směřovaný k zachování všech funkčností od začátku vývoje až po finální verzi programu. Tím je myšleno, že je aplikace schopna vygenerovat kompletní trénovací data, ale i dataset s doménami z různých zdrojů nebo jen pro jednotlivé vlastnosti entit. Rozhodnutí, jaký dataset trénovacích dat má být vygenerován, je řízeno z hlavní funkce pomocí *switch* příkazu, který zpracovává vstupní argumenty. Součástí hlavní funkce je i možnost spuštění pomocné analytické funkce, a to rovněž použitím vhodného argumentu. I když je aplikace Tool implementována pouze jako podpůrný nástroj, který nutně nemusí mít uživatelské rozhraní, rozhodl jsem se pro ovládání programu alespoň

pomocí vstupních argumentů, aby byla urychlena práce při experimentálním vyhodnocování všech datasetů.

Vstupní argumenty jsou dvě číselné hodnoty, z nichž první určuje, zda se jedná o entity domén mnou nalezených nebo entity z Featured Articles, anebo zda se má vykonat pomocná analytická funkce pro predikáty. Druhý argument pak určuje, které vlastnosti se mají z entit extrahovat, případně zda se mají extrahovat všechny. Konkrétnější a detailnější popis argumentů, ale i instalace a spuštění aplikace Tool, lze nalézt v příloze D.

3.2.7 Konfigurační parametry

Ke snadné instalaci a spuštění jakékoliv aplikace patří možnost uživatelsky nastavit důležité parametry pro její běh. K přidání jednoduchého systému nastavování parametrů do aplikace Tool bylo rozhodnuto hlavně z důvodu její závislosti na HDT souboru, který obsahuje DBpedia RDF data. Dalším důvodem je, že aplikace generuje trénovací datasety různých typů, ale i analytická data. Proto je nejčastější hodnotou nastavení parametru cesta ke zdrojovému souboru nebo názvy složek a souborů, které budou použity při ukládání výstupních dat. Detailní popis všech možných nastavení je uveden v příloze D. K implementaci správy parametrů byla použita třída *java.util.Properties*[75].

3.2.8 Zálohování

Účel pojmu zálohování ve spojení s aplikací Tool nemusí být hned na první pohled zřejmý. V tomto oddíle bude vysvětleno, proč se má zálohování provádět a jaká data (informace) má aplikace Tool zálohovat.

Z počátku vývoje této aplikace byla zpracovávána data pouze s malým počtem domén a entit, ovšem když aplikace pokročila do fáze zpracování všech domén Featured Articles, celý proces se výrazně zpomalil. Hlavní příčinou pomalého běhu aplikace bylo získávání informací o vlastnostech entit (predikáty, *rdf:type* a *dct:subject*) z HDT souboru pomocí SPARQL dotazů. Proto bylo navrženo řešení, které by tento proces urychlilo. Jelikož proces obnovy (změny) dat DBpedia entit a také domén Featured Articles není častý, usoudil jsem, že vhodným řešením tohoto problému bude vytvoření záloh domén, jejich příslušných entit, ale i jejich vlastností. Takto uložená data bude poté možné kdykoliv obnovit a rovnou zpracovat.

Formátem pro uložení dat byl zvolen JSON. Byl vybrán kvůli tomu, že se hodí a je navržený k uchování strukturovaných objektů, jako je například třída *Domain*. Jako nástroj pro ukládání a načítání záloh ze souborů JSON byl použit *JSON.simple* popsaný v oddíle 3.1.6. V návaznosti na předchozí oddíly bylo využívání záloh jednou z hlavních příčin použití konfiguračních parametrů a vstupních argumentů, aby bylo možné uživateli nabídnout jak

[75] Dostupné online: `https://docs.oracle.com/javase/7/docs/api/java/util/Properties.html` (26. 3. 2016).

možnost načtení dat ze záloh, tak i přímou extrakci dat z datasetu DBpedie. Pomocí konfiguračních parametrů lze poté nastavit, do jaké složky zálohy ukládat a jaké názvy souborů používat.

Tato funkcionalita byla použita i při uložení informací pouze o doménách a jejich entitách, z důvodu snadné identifikace zdrojů vytvořených trénovacích datasetů pro další uživatele aplikace.

Příklad dat uložených pro potřebu zálohování ve formátu JSON vypadá takto:

```
{
  "name": "food",
  "lowerBound": 0.1,
  "upperBound": 0.25,
  "entities": {
    "http://dbpedia.org/resource/Fish_and_chips": {
      "predicates": [
        " http://dbpedia.org/ontology/abstract",
        " http://dbpedia.org/ontology/alias",
        " http://dbpedia.org/ontology/ingredient",
        " http://dbpedia.org/ontology/origin",
        " http://dbpedia.org/ontology/thumbnail",
        ...
      ],
      "dcterms:subjects": [
        " http://dbpedia.org/resource/Category:Fast_food",
        " http://dbpedia.org/resource/Category:Fish_dishes",
        ...
      ],
      "rdf:types": [
        " http://dbpedia.org/ontology/Food"
      ]
    },
    ...
  }
}
```

Na příkladě lze vidět část zálohy domény *Food*. Tři tečky (...) vyjadřují zkrácení textu (pro potřeby této práce), ten v původním souboru zálohy pokračuje. Kompletní soubory se zálohami lze nalézt na přiloženém CD nebo ve veřejném repozitáři z podkapitoly 3.4.

3.2.9 Rozšíření domén s malým počtem entit

Jak vyplývá z názvu oddílu, tato část vývoje aplikace se týká fungujícího algoritmu. Problém domén s malým počtem entit je dán informacemi z Featured

Articles a naráží na to, že počty entit v jednotlivých doménách jsou nevyvážené. Proto při klasifikaci dochází k tomu, že některé domény mají oproti ostatním hodně nízkou až nulovou hodnotu *F-measure*. Proto jsem se této problematice rozhodl věnovat a navrhl jsem a implementoval jedno řešení, které zde bude blíže popsáno.

Z počátku celého procesu byly analyzovány počty entit ve všech doménách, aby bylo možné co nejpřesněji určit rozsah problému a zvolit tak domény, které je nutné nějakým způsobem rozšířit. Výsledkem bylo zjištění, že nejvíce entit o počtu 290 má doména *Media*, nejméně entit mají domény *Royalty and nobility* a *Mathematics* s počty 8, respektive 9. Průměrný počet entit v doménách byl 93, 6. Tato práce se ovšem zaměřila na domény s počtem entit menším než 20. Těchto domén bylo celkem 7:

- *Royalty and nobility*
- *Mathematics*
- *Philosophy and psychology*
- *Business, economics, and finance*
- *Language and linguistics*
- *Food and drink*
- *Computing*

Z názvů těchto domén není možné přesněji určit bližší příčinu malého počtu entit. Nelze říci, že jsou to domény s malým počtem pojmenovaných entit v celé Wikipedii. Je možné, že u těchto domén není od autorů a editorů kladen důraz na kvalitu a formu vytváření článků tak, aby splňovaly pravidla Featured Articles.

V dalším kroku bylo nutné určit, jakým způsobem budou domény rozšiřovány. Jednou z možností bylo data rozšířit ručním vyhledáváním podobně jako v počátku vývoje této aplikace. Cílem ovšem bylo spíše navrhnout automatický proces založený například na heuristice. V tomto procesu se dá vycházet z informací, které už byly analyzovány a použity při vytváření trénovacích datasetů. Práce se zaměřila na informace z *rdf:type*. Z podkapitoly 2.3 už víme, že *rdf:type* jsou třídy popisující danou entitu, které se extrahují z Wikipedia infoboxů. Proto může být vhodné je využít právě pro získání podobných entit z dané domény. Toto získání bylo navrženo tak, že z již známých entit se vyberou kombinace *rdf:type* a použijí se pro vytvoření nového SPARQL dotazu. Ze získaných dat se poté doplní doména novými entitami tak, aby byl výsledný počet roven 30. Pro hodnotu 30 jsem se rozhodl proto, aby se doplnilo co nejvíce entit, ale na druhou stranu, aby data zůstala relevantní své doméně.

Rozšíření byla implementována vytvořením pomocné třídy *DomainExtension*, která nejdříve zpracuje všechny entity z předložené domény a z jejich

rdf:type vytvoří seznam kombinací. Poté vytvoří podobný dotaz jako v oddíle 3.2.2 s tím rozdílem, že vkládaná proměnná může obsahovat i více identifikátorů tříd *rdf:type* najednou. Dotaz ve výsledku vypadá takto:

```
String queryString1 =
  "PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>" +
  "SELECT DISTINCT ?subject " +
  "where { " +
  "?subject rdf:type "+ s +" . " +
  "}";
```

Proměnná *s* je zřetězený seznam *rdf:type* pro právě vybranou entitu. Toto dotazování probíhá v cyklu pro všechny entity vybrané domény.

3.2.9.1 Shrnutí a diskuze k oddílu rozšíření domén

Rozšíření o nové entity se podařilo pro 5 ze 7 domén. Implementovaná heuristika nefungovala pro domény *Royalty and nobility* a *Mathematics*. Po analýze těchto dvou domén bylo zjištěno, že entity, které obsahují, nemají žádné *rdf:type*, proto ani nebylo možné provést dotazy k rozšíření. U zbylých domén, kde proces proběhl, lze sledovat mírné zlepšení hodnot *F-measure*. Porovnání původních a rozšířených modelů bude probráno v kapitole 4.

Byl proveden pokus o rozšíření domén s malým počtem entit, který nedopadl úplně podle očekávání, ale neúspěšný také nebyl. Metod rozšíření domén je veliké množství a mohou být založeny na různých algoritmech, heuristikách nebo zdrojích dat. Tento pokus o vytvoření alespoň jedné metody může být motivací pro další pokračování ve vývoji této aplikace i mimo rozsah této práce.

3.3 REST API

Jakmile byly úspěšně vygenerovány trénovací datasety a z nich natrénovány první modely, začal návrh REST API. Stejně jako v předchozí podkapitole 3.2 bude i tento text rozdělen podle fází vývoje této aplikace. V první řadě bylo nutné zajistit, aby REST API správně přijímala požadavky. Poté aplikace musela zpracovat požadavek a získat informace o předložené entitě, které bude používat při vyhodnocování predikce nad natrénovaným modelem. Konečnou fází vývoje je poté odeslání výsledků zpět klientovi.

3.3.1 Seznámení s knihovnou JAX-RS

Knihovna *JAX-RS* byla představena v oddíle 3.1.5, v této části bude představena podrobněji v kontextu REST API. Společnost *Oracle* vytvořila k této knihovně velmi kvalitní dokumentaci, která pomohla při jejím využívání. Jak již bylo zmíněno, knihovna používá k definování REST akcí anotace, které

se zapisují textem se „zavináčem“ (@) na začátku a vždy o řádek výše před definici třídy nebo metody, kterou má popisovat. Například definice cesty ke zdroji lze vyjádřit takto:

```
@Path("resources/")
```

Aby knihovna mohla spravovat všechny zdroje a ty pak předložit webovému serveru, musí aplikace obsahovat třídu *ApplicationConfig*, ve které je definovaná aplikační cesta (*ApplicationPath*[76]). Dále musí obsahovat metodu *getClasses()*, jejíž návratovou hodnotou je seznam tříd, které tvoří zdroje. Z této konfigurace plyne, že jakýkoliv zdroj bude spravován svojí třídou, kterou stačí pouze zaregistrovat ve výše popsané konfiguraci.

3.3.2 Zpracování a vyhodnocení požadavku

V konkrétním návrhu REST API se bude ke zdroji (predikcím domén pojmenované entity) přistupovat přes URI definovanou jako „/entities/{entity}-/domains“. To znamená, že aplikační cestou je zde „/entities“ a zbytek cesty „{entity}/domains“ vede ke třídě *EntitiesResource*, která provádí zpracování požadavku a celý proces vyhodnocení. Při takto definované URI se musíme vrátit k výrazu „{entity}“. Ten v aplikaci znamená, že součástí URI je proměnná, kterou lze zpracovávat. Konkrétně se jedná o identifikátor DBpedia pojmenované entity.

Jakmile dorazí požadavek s tímto identifikátorem, přijdou na řadu porovnání dalších aspektů, která rozhodnou o dalších krocích. Těmito aspekty jsou HTTP metoda a hlavičkový parametr *Accept*, které jsou oba rovněž definovány pomocí anotací. S ohledem na jednoduchost REST API této práce je využívána pouze HTTP metoda *GET*. Ovšem parametr *Accept*, jenž vyjadřuje formát dat odpovědi, může nabývat hodnot buď *text/plain*, nebo *application/json*. Hlavním a doporučovaným formátem odpovědi je JSON[77], a to ze stejných důvodů, jaké byly popsány v předchozí kapitole. Formát jednoduchého textu byl zvolen jako pomocný. Oba dva formáty jsou zpracovávány různými třídními metodami, ovšem je to jen z důvodu vytvoření jiné formy odpovědi. Tímto se dostáváme ke zpracování vstupní pojmenované entity a vyhodnocení predikce domén.

Zpracování vstupní pojmenované entity obstarává třída *Entity*, která byla použita i v aplikaci Tool, s přidáním automatické extrakce vlastností entity (predikáty, *rdf:type* a *dct:subject*). K procesu vyhodnocení predikce používá třída *EntitiesResource* čtyři metody. Každá zpracovává jiné vlastnosti entity, a proto používá jiný model. Používají se modely, které byly vygenerovány aplikací Tool, a jednotlivé metody s nimi pracují na stejném principu. Hlavní myšlenka vyhodnocení predikce spočívá v tom, že je vstupní entita vložena do

[76] Anotace: @javax.ws.rs.ApplicationPath("/").
[77] HTTP hlavička: „Accept: application/json“.

modelu jako jeho instance a poté je nad touto instancí provedena klasifikace. Aby bylo možné tento proces úspěšně vykonat, je nutné aby instance (entita) vyplnila celý datový řádek (přeneseno na soubor ARFF z oddílu 1.2.2) kromě třídy (domény), která jí poté bude predikována. Proto se v počáteční fázi načítá nejen model, ale i jemu příslušný ARFF soubor, z něhož jsou extrahovány všechny atributy (vlastnosti) a na jejich základě je vytvořena instance. Tato instance je poté předložena modelu a výsledkem jeho klasifikace jsou dvě hodnoty: predikovaná doména a procenta, s jakou pravděpodobností entita do domény patří.

Po vyhodnocení všech čtyř metod je vytvořena odpověď v požadovaném formátu a odeslána zpět klientovi. Příklad odpovědi ve formátu JSON:

```
{
    "predicates": {
        "domain": "Sport_and_recreation",
        "percents": "97,54"
    },
    "all_info": {
        "domain": "Sport_and_recreation",
        "percents": "100,00"
    },
    "dcterms:subjects": {
        "domain": "Media",
        "percents": "93,59"
    },
    "rdf:types": {
        "domain": "Sport_and_recreation",
        "percents": "39,03"
    },
    "url": "http://dbpedia.org/resource/Ice_hockey"
}
```

Příklad odpovědi ve formátu jednoduchého textu:

```
Entity: http://dbpedia.org/resource/Ice_hockey
predicates: Sport_and_recreation 97,54%
rdf:types: Sport_and_recreation 39,03%
dcterms:subjects: Media 93,59%
all_info: Sport_and_recreation 100,00%
```

3.3.3 Konfigurační parametry

Z důvodu dotazování se nad vstupní entitou, podobně jako v aplikaci Tool, bylo nutné použít napojení na DBpedia dataset. To bylo hlavním důvodem vytvoření podobné konstrukce správy konfiguračních parametrů jako v oddíle

3.2.7. Jelikož při vyhodnocování predikce domén vycházíme z toho, že aplikace může použít vždy pouze jeden model, je nutné před jejím spuštěním zvolit dvě konfigurace modelu. Z tohoto důvodu se zde nabízela možnost toto nastavení rovněž zahrnout mezi konfigurační parametry.

První z konfigurací je zdroj trénovacích datasetů a tedy typy domén, na kterých je prováděno učení. Druhá konfigurace určuje klasifikátor, pomocí kterého byl model vytvořen. Pro obě dvě nastavení lze vždy vycházet ze čtyř možností, které budou lépe popsány a porovnány v další kapitole 4. Detailní popis konfigurace a spuštění aplikace je popsán v příloze E.

3.4 Veřejný repozitář

Z podkapitoly 3.1 lze odvodit, že při návrhu a implementaci obou aplikací byly použity hlavně open-source nástroje. To přispělo k rozhodnutí, že výsledek této práce (obě aplikace) bude zveřejněn na webu ve veřejném repozitáři. Dalším důvodem byla možnost nabídnout aplikace k využití v komunitách DBpedia a Wikipedia.

Projekt s oběma aplikacemi je uložen na serveru *Bitbucket*[78], který slouží ke správě projektů a nabízí použití verzovacích nástrojů, jako je například *GIT*[79] nebo *Mercurial*[80]. Projekt je zveřejněný pod licencí GNU GPLv3[81] a je dostupný na veřejné URL adrese[82]. V tomto repozitáři jsou uložené nejen zdrojové kódy obou aplikací, ale i analýzy, natrénované modely a výsledky experimentů.

3.5 Ukázka použití

Příklad použití nástroje automatického učení domén entit může být předveden ve spojení s nástrojem DBpedia Spotlight[83]. Tento nástroj slouží k automatické anotaci DBpedia entit v textech, pomocí něhož budou získány identifikátory DBpedia entit a poté předloženy REST API, která pro tyto entity provede automatické učení domén.

V příkladu byl použit jeden odstavec textu z Wikipedia, který lze vidět v následující citaci:

> More injuries struck Jágr in the **2006 Winter Olympics** in **Turin**. He was injured after a hit from **Finland**'s **Jarkko Ruutu**,

[78] Více informací online: `https://bitbucket.org/` (25. 4. 2016).

[79] Více informací online: `https://git-scm.com/` (25. 4. 2016).

[80] Více informací online: .`https://www.mercurial-scm.org/` (25. 4. 2016).

[81] GNU GPLv3 – GNU General Public License Version 3 – Dostupné online: `http://www.gnu.org/licenses/gpl.html` (26. 4. 2016).

[82] Dostupné online: `https://bitbucket.org/benaktom/learningdomainsfornamedentities/` (30. 4. 2016).

[83] Více informací online: `https://github.com/dbpedia-spotlight/dbpedia-spotlight/wiki/Introduction` (4. 5. 2016).

Tabulka 3.1: Ukázka použití automatického učení domén entit ve spojení s nástrojem DBpedia Spotlight.

Identifikátor pojmenované entity	**Predikovaná doména**	**Procenta**
2006_Winter_Olympics	Sport and recreation	73,00%
Finland	Geography and places	97,00%
Jarkko_Ruutu	Culture and society	62,24%
Muscle	Physics and astronomy	15,71%
2010_Winter_Olympics	Sport and recreation	100,00%
Ice_hockey_at_the_2002-_Winter_Olympics	Sport and recreation	100,00%
Czech_Republic	Geography and places	99,00%
2014_Winter_Olympics	Sport and recreation	100,00%
Sochi	Geography and places	74,00%
Vysvětlivky: Řetězci ID pojm. entity předchází URL http://dbpedia.org/resource/. Procenta – s jakou pravděpodobností entita do domény patří.		

> requiring stitches to his eyebrow. The injury, however, was not as serious as first anticipated, and Jágr was able to play in the following games, though he was unable to finish the bronze medal game due to **muscle** injury. Despite the trouble, Jágr won his second-career Olympic medal, the bronze. In 2010, Jágr was his nation's flag bearer at the **2010 Winter Olympics**, but in the men's **ice hockey** tournament, the **Czechs** finished a disappointing seventh after a defeat in the quarterfinals to **Finland**. Jágr again represented his country at the **2014 Winter Olympics** in **Sochi**, scoring two goals and one assist in five games as the **Czech Republic** again lost in the quarterfinals.

V textu jsou tučným písmem zvýrazněné pojmenované entity tak, jak je anotoval nástroj DBpedia Spotlight. Následující tabulka 3.1 obsahuje výsledky predikcí domén provedených nástrojem REST API.

Kapitola 4

Experimenty

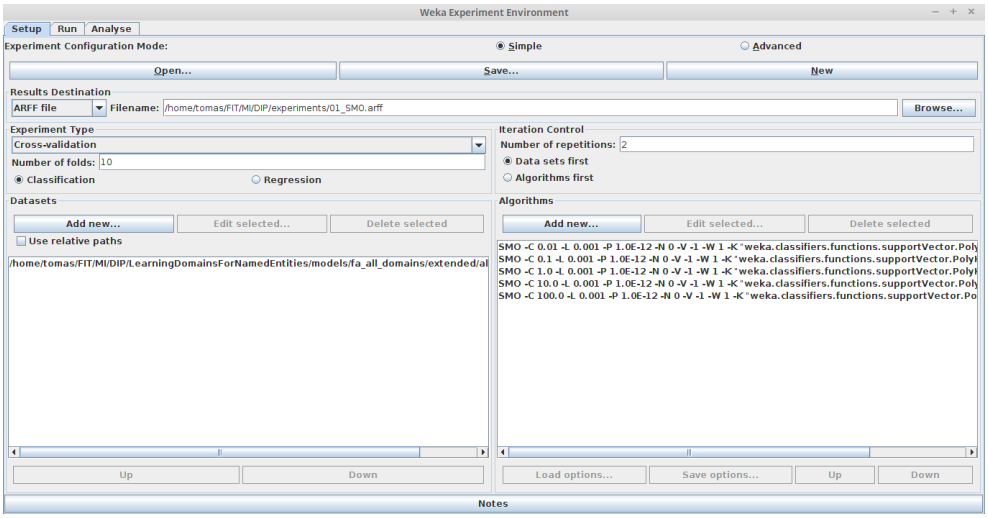
V této kapitole budou představeny experimenty provedené na datasetech trénovacích instancí, které byly vygenerované aplikací Tool. Experimenty jsou rozděleny do několika skupin podle zaměření dané problematiky trénování. V následujících podkapitolách budou nejdříve popsány cíle experimentů, poté představeno nastavení a datové rozdělení a nakonec bude provedeno zhodnocení výsledků a vyvozen závěr.

4.1 Cíle

Cílem experimentů je nalézt nejoptimálnější trénovací data a klasifikátory k učení domén pojmenovaných entit. Z provedených experimentů by měly být vyvozeny výsledky, které prokáží, zda testovaná data nebo algoritmy jsou vhodné nebo nevhodné pro danou problematiku. Hlavním cílem je také určit nejvhodnější datasety, které budou využívány REST API v jejím produkčním nasazení.

4.2 Konfigurace experimentů

Nutným faktem pro provádění kvalitních experimentů je uvedení přesného nastavení a rozdělení experimentů do správných skupin, aby byla výsledná data relevantní k porovnávání mezi sebou. Experimenty budou rozděleny do tří skupin, které se budou zajímat o různé aspekty problému učení domén. Nejprve budou provedeny experimenty nad různými druhy algoritmů klasifikace a poté se budou testovat data z pohledu zdrojů domén a entit a z pohledu vlastností entit. Experimenty budou prováděny pomocí programu Weka, který nabízí pro toto testování celé rozhraní nazvané *Experiment Environment*. Toto rozhraní je schopné po provedení experimentů porovnat výsledky podle různých metrik. Hlavními metrikami, popsanými v oddíle 1.2.4, budou *Precision*, *Recall* a *F-measure*. Také je nutné dodat, že k evaluaci byla použita technika



Obrázek 4.1: Ukázka grafického rozhraní nástroje Weka Experiment Environment. Pořízeno 3. 5. 2016 jako snímek obrazovky spuštěného programu.

Cross-validation, která dataset rozdělovala na 10 částí. Celý proces byl pro každý experiment dvakrát opakován. Náhled prostředí *Weka Experiment Environment* lze vidět na obrázku 4.1, který navíc zobrazuje počáteční nastavení experimentu pro algoritmus SMO (viz následující oddíl 4.2.1).

4.2.1 Algoritmy klasifikace

Jak bylo popsáno v podkapitole 2.5, k natrénování modelů byly vybrány čtyři klasifikační algoritmy[84]:

- *Naive Bayes*
- *Sequential Minimal Optimalization (SMO)*
- *k-Nearest Neighbours (kNN)*
- *Random Forest*

V těchto experimentech budou porovnávány všechny algoritmy mezi sebou. Jelikož nelze skupiny experimentů testovat najednou, bude zvolen jeden nejvhodnější dataset a na něm provedeny všechny klasifikace. Tento dataset se bude skládat z dat rozšířených domén Featured Articles se všemi sjednocenými vlastnostmi, pracovním názvem *fa_all_ext_domains_merged*. Dále je nutné brát v úvahu různá nastavení parametrů jednotlivých algoritmů. I toto bude předmětem těchto experimentů. V následujících pododdílech budou popsány parametry nastavující jednotlivé klasifikační algoritmy.

[84]Názvy jsou uvedeny v anglickém tvaru.

4.2.1.1 Naive Bayes

Algoritmus Naive Bayes nepoužívá při trénování žádné významné parametry, což je dáno jeho podstatou popsanou v oddíle 2.5.3.

4.2.1.2 Sequential Minimal Optimalization (SMO)

SMO je algoritmus, který byl popsán v oddíle 2.5.5 a kterému při experimentování bude nastaven 1 parametr, a to:

- **C** – Parametr volby komplexnosti, který SMO využívá k vytvoření nadroviny mezi dvěma třídami. Pro experimenty byly zvoleny hodnoty 0.01, 0.1, 1, 10, 100.

Dalším parametrem, který by mohl být nastavován, je volba jádrové funkce. Pro potřebu této práce a těchto experimentů bude však využito jen jádro polynomiální s exponentem nastaveným na hodnotu 1. Vzorec tohoto jádra je v programu Weka uváděný jako $K(x, y) =< x, y >^p$ $orK(x, y) = (< x, y > +1)^p$. Jiné možnosti tohoto nastavení nebyly použity, protože jejich výpočetní čas byl moc dlouhý[85].

4.2.1.3 k-Nearest Neighbours (kNN)

U algoritmu kNN popsaného v oddíle 2.5.4 se dají nastavovat 3 parametry. Jsou to:

- **K** – Parametr nastavující, kolik nejbližších sousedních instancí se má použít. Pro experimenty byly zvoleny hodnoty 1, 2 a 3.

- **Vzdálenostní funkce** – Parametr určující vzdálenostní funkci, která je použita pro výpočet vzdálenosti mezi instancemi. Pro experimenty byly zvoleny funkce *EuclideanDistance* a *ManhattanDistance*.

- **Váha vzdálenostní funkce** – Parametr upravující vypočtenou vzdálenost vážením. Pro experimenty byla nastavena váhová funkce na $\frac{1}{d}$, $1-d$ nebo *bez vážení* (d je vypočtená vzdálenost).

Kromě výše uvedených vzdálenostních funkcí bylo možné využít funkce *ChebyshevDistance* a *EditDistance*, ty ovšem nebyly využity kvůli příliš dlouhému výpočetnímu času[86].

[85] Při testování jiného nastavení jádrové funkce nebyla klasifikace dokončena, protože výpočetní čas trval déle než 2 hodiny (na dostupném zařízení zvoleném k experimentům popsaném v podkapitole 4.3).

[86] Při testování těchto vzdálenostních funkcí nebyla klasifikace dokončena, protože výpočetní čas trval déle než 2 hodiny (na dostupném zařízení zvoleném k experimentům popsaném v podkapitole 4.3).

4.2.1.4 Random Forest

Algoritmu Random Forest popsanému v oddíle 2.5.1 lze nastavit 3 parametry:

- **Počet stromů** – Parametr, který určuje, kolik má být vygenerováno náhodných stromů. Pro experimenty byly zvoleny hodnoty 50, 100 a 150.

- **Maximální hloubka stromu** – Parametr nastavující maximální hloubku stromů. Lze počítat i s neomezenou hloubkou (nastavením parametru na 0). Pro experimenty byly zvoleny hodnoty 0 (neomezeně), 10, 20, 30, 40, 50, 100 a 150.

- **Počet vlastností (atributů)** – Parametr určující počet atributů, které budou použity při jednom náhodném výběru. Pro experimenty byly vybrány hodnoty 0 (automatický výběr), 10, 20, 30, 40.

4.2.2 Data

Pojmem data z nadpisu toho oddílu jsou myšleny zdroje entit a jejich vlastnosti, které byly použity k trénování. Tato část experimentu se dá dále rozdělit na experimenty nad datasety a experimenty nad vlastnostmi entit.

4.2.2.1 Datasety

Na první pohled jsou datasety rozděleny na domény s mnou vyhledanými entitami a domény s entitami z Featured Articles. Ovšem s takovýmto rozdělením nelze datasety jednoduše porovnat. Proto z pohledu více do hloubky lze data rozdělit do dvou skupin se dvěma porovnatelnými datasety.

V prvním experimentu budou testovány datasety, které vznikly na počátku vývoje aplikace Tool. Lze tedy říci, že půjde o experimenty porovnávající entity nalezené mnou a entity z Featured Articles. Protože domény s mnou nalezenými entitami jsou pouze tři, budou porovnávány s doménami stejných názvů, ovšem s entitami z Featured Articles. Jedná se o domény *Sport and recreation*, *Music* a *Food and drink*. Větší počet domén nelze takto srovnávat, protože není k dispozici více domén s mnou vyhledanými entitami, což také nebylo účelem této práce.

Druhý experiment bude zaměřený na všechny domény z Featured Articles a jejich entity. Jak bylo popsáno v 3.2.9, aplikace Tool je schopna provést jednoduché rozšíření domén s malým počtem entit. Z tohoto faktu plyne, že experiment bude obsahovat porovnání datasetů domén Features Articles s původním počtem entit a s počtem rozšířeným.

Oba experimenty budou vycházet s nejvhodnějším nastavením z předchozího oddílu 4.2.1.

4.2.2.2 Vlastnosti entit

Posledním nastavením experimentů je porovnání datasetů podle vlastností entit, na které se zaměřují. Tyto vlastnosti byly popsány v 2.3, a jsou to: predikáty, *rdf:type* a *dct:subject*. Navíc s nimi bude porovnán dataset, který je vytvořen z kombinace všech těchto vlastností, zde pracovně nazývaný jako *merged*. Experimenty budou provedeny vždy na nejvhodnějším datasetu z každé skupiny z předchozího oddílu 4.2.2 a s nejvhodnějším algoritmem z oddílu 4.2.1.

4.3 Popis prostředí

Experimenty byly prováděny na osobním notebooku s touto charakteristikou:

- Notebook Lenovo E430
- Procesor Intel Core i5-3210M CPU @ 2.50GHz x 2
- Paměť 11.3 GiB
- Systém Linux Mint 17.1 Cinnamon 64-bit
- Jádro Linuxu 3.13.0-37-generic

4.4 Vyhodnocení experimentů

V této podkapitole budou představeny a popsány výsledky jednotlivých experimentů. Text této kapitoly bude rozdělen stejně jako v podkapitole 4.2 na dva oddíly.

4.4.1 Experiment: algoritmy klasifikace

Výsledky experimentu zaměřeného na porovnání algoritmů klasifikace jsou znázorněny v tabulce 4.1. Tabulka obsahuje úzký výběr ze všech naměřených výsledků. Kompletní výsledky rozdělené do tabulek po jednotlivých algoritmech lze nalézt v příloze F.

Na první pohled lze spatřit, že nejhůře dopadl algoritmus Naive Bayes a nejlépe algoritmus Random Forest, jehož úspěšnost byla jen o málo lepší než u algoritmu SMO.

Malá úspěšnost algoritmu Naive Bayes může být dána nevyvážeností četnosti výskytu atributů a tříd, tedy nevyvážeností počtu trénovacích instancí jednotlivých domén (tříd).

O zhruba desetinu lepších výsledků dosáhl algoritmus kNN. U tohoto algoritmu se ukázalo, že výsledky při použití Euklidovy vzdálenostní funkce nebo Manhattanské funkce jsou totožné. Dále můžeme sledovat určité zlepšení (o jednu setinu) při použití vážení vzdálenostních funkcí, ovšem toto zlepšení

Tabulka 4.1: Experiment: algoritmy klasifikace (úspěšnost).

Algoritmus	**Precision**	**Recall**	**F-measure**
Naive Bayes	0.65	0.57	0.56
SMO C=0.1	0.77	0.69	0.70
SMO C=1.0	0.75	0.69	0.69
kNN k=3, vzdál.f.=EuclDist, váha=1/d	0.71	0.65	0.65
kNN k=2, vzdál.f.=EuclDist, váha=1/d	0.72	0.65	0.66
kNN k=1, vzdál.f.=EuclDist, váha=no	0.72	0.66	0.66
kNN k=3, vzdál.f.=ManhDist, váha=1-d	0.71	0.64	0.65
kNN k=2, vzdál.f.=ManhDist, váha=1-d	0.72	0.65	0.66
Random Forest strom=50, hl.=30, vlast.=auto	0.75	0.67	0.67
Random Forest strom=100, hl.=inf, vlast.=20	0.74	0.70	0.70
Random Forest strom=150, hl.=40, vlast.=auto	0.76	0.70	0.70
Random Forest strom=150, hl.=inf, vlast.=auto	0.75	0.71	0.71
Random Forest strom=150, hl.=100, vlast.=15	**0.76**	**0.72**	**0.72**

Vysvětlivky:
vzdál.f. – Vzdálenostní funkce.
EuclDist – *EuclideanDistance*.
ManhDist – *ManhattanDistance*.
strom – Počet stromů.
hl. – Maximální hloubka stromu.
vlast. – Počet vlastností (atributů).

je znatelné pouze u algoritmů s $k > 1$. Pokud se zaměříme na hodnotu k, můžeme sledovat, že při jeho zvětšování se výsledné hodnoty *F-measure* zhoršují. Z takto naměřených hodnot by se mohlo zdát, že nejlepším nastavením algoritmu kNN je jeho speciální případ, tedy NN.

U algoritmu SMO můžeme sledovat, že má nejlepší hodnotu přesnosti (*Precision*), ale oproti tomu má výrazně horší hodnotu citlivosti (*Recall*), a proto jeho výsledná *F-measure* zůstává o kousek horší než u algoritmu Random Forest. Dále je vidět, že parametr komplexnosti (C) výrazně ovlivňuje výslednou hodnotu *F-measure*.

Pro algoritmus Random Forest bylo provedeno nejvíce experimentů s nastavením parametrů. To bylo dáno tím, že každému parametru bylo možné nastavit velké množství hodnot. Nastavování těchto parametrů probíhalo ve třech iteracích vždy podle typu nastavovaného parametru z pododdílu 4.2.1.4. Ve čtvrté iteraci byly zkombinovány hodnoty nastavení z předchozích iterací, které byly uznané jako vhodné k otestování. Z této čtvrté iterace vzešlo nejvhodnější nastavení algoritmu Random Forest s výslednou hodnotou *F-measure* = 0.72.

Dalším aspektem, který byl v tomto experimentu sledován, byl výpočetní čas algoritmů při trénování a při testování. V tomto ohledu je nutné zmínit, že

Tabulka 4.2: Experiment: algoritmy klasifikace (výpočetní čas).

Algoritmus	**Training**	**Testing**
Naive Bayes	0.86	4.04
SMO C=0.1	20.17	1.14
kNN k=2, vzdál.f.=EuclDist, váha=1/d	0.00	12.66
Random Forest strom=150, hl.=100, vlast.=15	30.60	0.12
Vysvětlivky: Training – User CPU time training – Výpočetní čas trénovaní v sekundách. Testing – User CPU time testing – Výpočetní čas testování v sekundách. Ostatní zkratky jsou stejné jako v tabulce 4.1.		

nejlépe dopadl algoritmus kNN, jehož výpočetní čas trénování je téměř nulový. Jednotlivé výpočetní časy lze vidět v tabulce 4.2, ve které je pouze výběr z algoritmů. Celá tabulka výpočetních časů algoritmů je uvedena v příloze F.

Ve všech hlavních ohledech popsaných v tomto oddíle můžeme dojít k závěru, že pro učení domén pojmenovaných entit s trénovacími instancemi vytvořenými nad Featured Articles je nejvhodnějším algoritmem Random Forest s tímto nastavením parametrů: Počet stromů = 150, Maximální hloubka stromu = 100, Počet vlastností (atributů) = 15. Z tohoto důvodu byl algoritmus zvolen pro experimenty v dalším oddíle 4.4.2.

4.4.2 Experiment: datasety a vlastnosti entit

Jak už bylo popsáno v oddíle 4.2.2, experimenty zaměřené na data byly rozděleny na skupinu zabývající se datasety a skupinu zabývající se vlastnostmi entit. Jelikož je možné tyto dvě skupiny spojit, byly výsledky experimentů kvůli přehlednosti rozděleny podle typu datasetů. Znázornění těchto výsledků lze nalézt v tabulkách 4.3 a 4.4.

První tabulka 4.3 je zaměřena na porovnání datasetů domén s mnou vyhledanými entitami (dále jen datasety *my*) a datasetů domén Featured Articles (dále jen datasety Featured Articles). Při celkovém pohledu na tyto datasety je znatelné, že jsou datasety Featured Articles úspěšnější, jejich hodnoty *F-measure* jsou lepší. To může být dáno tím, že datasety *my* jsou sice sestaveny z entit, jejichž názvy patří do zvolených domén, ale tyto entity nemusí obsahovat kvalitní strukturovaná data, což při prvním pohledu nelze snadno poznat.

Pokud se zaměříme na jednotlivé vlastnosti, nad kterými jsou datasety vybudovány, je vidět, že v obou případech figurují datasety postavené nad predikáty. U datasetů Featured Articles je pak dataset s predikáty vylepšen připojením datasetů ostatních vlastností (*rdf:type* a *dct:subject*) a vytvořen tak společný dataset (*merged*). V kontextu datasetu *merged* můžeme sledovat vylepšení, pouze pokud se hodnoty *F-measure* všech datasetů, ze kterých je složený, výrazně neliší. Například u datasetů *my* je vidět, že datasety posta-

Tabulka 4.3: Experiment: datasety a vlastnosti entit (1. část).

Dataset	Vlastnost	Precision	Recall	F-measure
my_3_domains	Predikáty	0.83	0.81	0.79
my_3_domains	*dct:subject*	0.52	0.57	0.49
my_3_domains	*rdf:type*	0.28	0.42	0.30
my_3_domains	*merged*	0.83	0.81	0.79
fa_3_domains	Predikáty	0.87	0.88	0.87
fa_3_domains	*dct:subject*	0.81	0.76	0.74
fa_3_domains	*types*	0.86	0.83	0.82
fa_3_domains	*merged*	0.88	0.89	0.88
Vysvětlivky: my_3_domains – Dataset 3 domén s mnou vyhledanými entitami. fa_3_domains – Dataset 3 domén Featured Articles.				

Tabulka 4.4: Experiment: datasety a vlastnosti entit (2. část).

Dataset	Vlastnost	Precision	Recall	F-measure
fa_all_domains	Predikáty	0.70	0.67	0.67
fa_all_domains	*dct:subject*	0.74	0.49	0.52
fa_all_domains	*rdf:type*	0.77	0.52	0.57
fa_all_domains	*merged*	0.74	0.71	0.70
fa_all_ext_domains	Predikáty	0.71	0.68	0.68
fa_all_ext_domains	*dct:subject*	0.74	0.49	0.51
fa_all_ext_domains	*rdf:type*	0.76	0.53	0.57
fa_all_ext_domains	*merged*	0.76	0.72	0.72
Vysvětlivky: fa_all_domains – Dataset všech domén Featured Articles. fa_all_ext_domains – Dataset všech domén Featured Articles s rozšířením domén s malým počtem entit.				

vené nad vlastnostmi *rdf:type* a *dct:subject* nijak nevylepšují dataset postavený nad predikáty.

Druhá tabulka 4.4 porovnává skupiny datasetů všech domén Featured Articles a jejich entit. První skupina datasetů (dále jen základní datasety) obsahuje domény a jejich entity přesně tak, jak jsou uvedeny ve Featured Articles. Druhá skupina (dále jen rozšířené datasety) je rozšířením základních datasetů popsaném v oddíle 3.2.9. Z tabulky lze vyčíst, že jsou podle očekávání úspěšnější rozšířené datasety, i když jen nepatrně. To je dáno tím, že metoda rozšíření domén s malým počtem entit se týkala jen 7 z 30 domén. Druhým důvodem může být použití jednoduché heuristiky v procesu rozšiřování.

Stejně jako u předchozích datasetů (*my* a Featured Articles) je v tomto případě jasně vidět, že dominují datasety postavené na predikátech. Podle

očekávání jsou datasety vytvořené spojením datasetů postavených nad všemi vlastnostmi nejúspěšnější.

Experimenty provedené v tomto oddíle potvrdily, že vybraný dataset *fa-_all_ext_domains_merged* pro experimenty z oddílu 4.4.1, je nejvhodnější pro učení domén pojmenovaných entit s trénovacími instancemi vytvořenými nad Featured Articles.

4.5 Shrnutí experimentů

Provedené experimenty pro algoritmy klasifikace s konfigurací z oddílu 4.2.1 ukázaly, že pokud uživatel klade důraz hlavně na kvalitu výsledků predikování domén pojmenovaných entit, je nejvhodnější zvolit algoritmus Random Forest s nastavením parametrů: Počet stromů = 150, Maximální hloubka stromu = 100, Počet vlastností (atributů) = 15. Ovšem pokud je preferováno zaměření spíše na výpočetní čas trénování, vhodným algoritmem je kNN s nastaveným parametrů: k = 2, Vzdálenostní funkce = *ManhattanDistance*, Váha vzdálenostní funkce = $\frac{1}{d}$.

Experimenty zaměřené na datasety a vlastnosti entit s konfigurací z oddílu 4.2.2 byly provedeny pomocí algoritmu Random Forest z předchozího odstavce. Z výsledků těchto experimentů můžeme vyvodit závěr, že vhodnějšími datasety zdrojových entit jsou datasety domén Featured Articles. U těchto datasetů je navíc dobré použít rozšíření domén s malým počtem entit s alespoň jednoduchou heuristikou (popsáno v oddíle 3.2.9). Při bližším zaměření na vlastnosti entit se ukázalo, že datasety postavené na predikátech byly úspěšným základem učení domén pojmenovaných entit. Celkovou úspěšnost těchto datasetů navyšovalo spojení s datasety ostatními (postavených na vlastnostech *rdf:type* a *dct:subject*), což potvrdilo vhodnost výběru datasetu při konfiguraci experimentů pro algoritmy klasifikace z oddílu 4.2.1.

Závěr

Cílem diplomové práce bylo navrhnout a implementovat nástroje k vygenerování modelů pro učení domén pojmenovaných entit a dále provést automatické učení nad těmito modely v podobě REST služeb. V první kapitole *Rešerše a vymezení pojmů* byly popsány základní pojmy týkající se této práce, jako jsou například sémantický web nebo strojové učení. Součástí této kapitoly byla i rešerše dosavadních řešení. V druhé kapitole byly provedeny analýzy existujících datových zdrojů pojmenovaných entit a domén. Dále bylo popsáno zadání a požadavky práce s představením pojmu REST služba a provedena analýza algoritmů klasifikace. Poznatky z těchto dvou kapitol byly využity ve třetí kapitole s názvem *Návrh a implementace*, která pojednává o použitých technologiích a popisuje postup návrhu a implementace aplikace na tvorbu trénovacích datasetů a REST API. Tato kapitola také obsahuje ukázky použití automatického učení domén pojmenovaných entit. V poslední čtvrté kapitole byly provedeny experimenty nad modely určenými k učení domén entit z druhé a třetí kapitoly této práce.

Výsledkem této diplomové práce je funkční systém, který je rozdělený na dvě aplikace. První aplikace vytváří trénovací datasety na základě různých zdrojů domén a vlastností entit. Z těchto datasetů jsou poté pomocí klasifikace vytvořeny natrénované modely, které jsou použity v druhé aplikaci. Ta provádí automatické učení domén pojmenovaných entit v podobě REST služeb. Obě aplikace naplnily funkční i nefunkční požadavky na systém popsané v podkapitole 2.2. Součástí výsledků této práce jsou i provedené experimenty, v jejichž vyhodnoceních jsou doporučeny algoritmy klasifikace, datasety a vlastnosti entit, které jsou nejvhodnější pro učení domén entit v závislosti na zvoleném zdroji pojmenovaných entit a domén.

Přínos práce

Přínosem této práce může být využití výsledné REST API s natrénovanými modely v nástrojích rozpoznávajících pojmenované entity (NER nástroje).

Tyto existující nástroje jsou vesměs generické a v procesu anotace vrací velký počet těchto entit. Uživatele však nemusí zajímat všechny entity, ale jen ty z dané domény. Pomocí REST API by mohl uživatel zredukovat počet nalezených pojmenovaných entit tím, že NER nástroj zaměří pouze na danou doménu, tedy provede vyfiltrování pojmenovaných entit podle určité domény.

Dalším přínosem mohou být provedené analýzy nacházející se v teoretické části práce. Tyto analýzy provádí rozbor existujících datasetů pojmenovaných entit a domén ve spojení se supervizovanými algoritmy strojového učení.

V neposlední řadě je možným přínosem vytvořená aplikace na tvorbu trénovacích datasetů, jež byla navržena a implementována tak, aby bylo možné provádět jednoduchá rozšíření z pohledu vlastností entit nebo použít různé zdroje datasetů pojmenovaných entit.

Možná rozšíření práce

Možným rozšířením této diplomové práce by mohlo být navázání na vývoj aplikace pro tvorbu trénovacích datasetů, a to konkrétně na funkcionalitu, která se zabývá rozšířením domén s malým počtem entit z oddílu 3.2.9. V této problematice se dá přímo zaměřit na návrh různých algoritmů a heuristik, které by byly schopné srovnat počty entit v doménách a zmenšit tak nevyváženost domén mezi sebou.

Dalším rozšířením, tentokrát zaměřeným na REST API, by mohla být integrace automatického učení domén entit přímo do NER nástrojů, jakými jsou například DBpedia Spotlight nebo Entityclassifier.eu[87]. Dále je také možné rozšířit automatické učení domén pojmenovaných entit o grafické uživatelské rozhraní.

[87] Více informací online: `http://entityclassifier.eu/thd/about/` (5. 5. 2016).

Literatura

[1] Titze, G.; Bryl, V.; Zirn, C.; aj.: DBpedia Domains: augmenting DBpedia with domain information. In *LREC*, 2014, s. 1438–1442.

[2] Berners-Lee, T.; Hendler, J.; Lassila, O.: The Semantic Web. *Scientific American*, 2001: s. 29–37.

[3] Bernes-Lee, T.: Linked Data. Dostupné online: `https://www.w3.org/DesignIssues/LinkedData/` (13. 3. 2016).

[4] Berka, P.: Strojové učení. Dostupné online: `http://sorry.vse.cz/~berka/docs/izi456/kap_4.pdf` (13. 3. 2016).

[5] open-source Meaning in the Cambridge English Dictionary. Dostupné online: `http://dictionary.cambridge.org/dictionary/english/open-source/` (13. 3. 2016).

[6] Paulheim, H.; Bizer, C.: Type Inference on Noisy RDF Data. In *The Semantic Web–ISWC 2013*, Springer, 2013, s. 510–525.

[7] domain Meaning in the Cambridge English Dictionary. Dostupné online: `http://dictionary.cambridge.org/dictionary/english/domain/` (13. 3. 2016).

[8] Ševčíková, M.; Žabokrtský, Z.; Krůza, O.: Zpracování pojmenovaných entit v českých textech. Technická Zpráva TR-2007-36, ÚFAL MFF UK, Prague, Czech Republic, 2007.

[9] Anderson, C.: The Long Tail. Dostupné online: `http://www.wired.com/2004/10/tail/` (13. 3. 2016).

[10] Wikipedia contributors: Wikipedia:About. Dostupné online: `https://en.wikipedia.org/wiki/Wikipedia:About/` (13. 3. 2016).

[11] crowdsourcing Meaning in the Cambridge English Dictionary. Dostupné online: http://dictionary.cambridge.org/dictionary/english/crowdsourcing/ (13. 3. 2016).

[12] Lehmann, J.; Isele, R.; Jakob, M.; aj.: DBpedia - A Large-scale, Multilingual Knowledge Base Extracted from Wikipedia. *Semantic Web*, ročník 6, č. 2, 2015: s. 167–195.

[13] Chen, E.: Choosing a Machine Learning Classifier. Dostupné online: http://blog.echen.me/2011/04/27/choosing-a-machine-learning-classifier/ (13. 3. 2016).

[14] Konsiantis, S. B.: Supervised Machine Learning: A Review of Classification Techniques. *Informatica*, ročník 31, 2007: s. 249–268.

[15] Liaw, A.; Wiener, M.: Classification and Regression by randomForest. *R News*, ročník 2/3, Prosinec 2002: s. 18–22.

[16] Friedman, N.; Geiger, D.; Goldszmidt, M.: Bayesian Network Classifiers. *Machine Learning*, ročník 29, č. 2-3, 1997: s. 131–163.

[17] Guo, G.; Wang, H.; Bell, D.; aj.: KNN Model-Based Approach in Classification. 2003, dostupné online: https://www.researchgate.net/profile/Gongde_Guo/publication/2948052_KNN_Model-Based_Approach_in_Classification/links/0fcfd50a47d7fcd0d8000000.pdf (13. 3. 2016).

[18] Eustaquio, R. G.; Karas, E. W.; Ribeiro, A. A.: Constraint Quailifications for Nonlinear Programming. *Federal University of Parana*, 2010, dostupné online: http://paginapessoal.utfpr.edu.br/eustaquio/my-research-interests/kkterabio.pdf (13. 3. 2016).

[19] Pairwise Coupling and Pairwise with Majority Voting. Dostupné online: http://svr-www.eng.cam.ac.uk/~kkc21/thesis_main/node25.html (13. 3. 2016).

[20] Platt, J. C.; aj.: Sequential Minimal Optimization: A Fast Algorithm for Training Support Vector Machines. 1998, dostupné online: http://www.msr-waypoint.com/pubs/69644/tr-98-14.pdf (13. 3. 2016).

[21] Vitvar, T.: MI-W20, Lecture 2: Representational State Transfer. Dostupné online: http://humla.vitvar.com/slides/w20/lecture2.html (13. 3. 2016).

[22] Fielding, R. T.: Architectural Styles and the Design of Network-based Software Architectures. Dostupné online: https://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm (13. 3. 2016).

[23] Novotný, L.: Historie a vývoj jazyka Java. Dostupné online: http://www.fi.muni.cz/usr/jkucera/pv109/2003p/xnovotn8.htm (3. 4. 2016).

Příloha **A**

Seznam použitých zkratek

API Application Programming Interface

ARFF Attribute-Relation File Format

CRUD Create, Retrieve, Update, Delete

DOM Document Object Model

GB Gigabyte

GNU GPL GNU General Public License

HDT Header, Dictionary, Triples

HTML HyperText Markup Language

HTTP Hypertext Transfer Protocol

JSON JavaScript Object Notation

JVM Java Virtual Machine

KKT Karush-Kuhn-Tucker

kNN k-Nearest Neighbours

LOD Linking Open Data

NER Named Entity Recognition

NN Nearest Neighbour

OWL Web Ontology Language

POM Project Object Model

RDF Resource Description Framework

REST Representational state transfer

SMO Sequential Minimal Optimization

SPARQL SPARQL Protocol and RDF Query Language

SVM Support Vector Machines

URI Uniform Resource Indentifier

URL Uniform Resource Locator

W3C World Wide Web Consortium

XML Extensible Markup Language

Příloha **B**

Obsah přiloženého CD

- `readme.txt` ..... stručný popis obsahu CD
- `src`
  - `application` ..... zdrojové kódy implementace
    - `REST_API` ..... zdrojové kódy REST API
    - `Tool` ..... zdrojové kódy aplikace na tvorbu trénovacích datasetů
  - `thesis` ..... zdrojová forma práce ve formátu LaTeX
- `text` ..... text práce
  - `DP_benaktom_2016.pdf` ..... text práce ve formátu PDF
- `miscellaneous` ..... ostatní soubory spojené s prací
  - `analysis` ..... provedené analýzy
  - `experiments` ..... provedené experimenty
  - `models` ..... trénovací datasety a modely

Příloha C

Analýza dat

Tabulka C.1: Analýza domén, datasetů a vlastností entit (1. část).

Doména	Počet entit	Horní mez	Spodní mez	Počet predikátů	Předpokládaná TaaS doména
Royalty and nobility	8	0.15	0.0	34	
Mathematics	9	0.34	0.1	22	2100 Natural sciences
Philosophy and psychology	12	0.34	0.16	48	
Business, economics, and finance	13	0.31	0.06	74	0300 Economics
Language and linguistics	13	0.47	0.2	29	2005 Social sciences
Food and drink	14	0.29	0.1	44	1000 Agriculture and foodstuff
Computing	16	0.57	0.17	32	
Geology and geophysics	23	0.4	0.08	74	2100 Natural sciences
Heraldry, honors, and vexillology	24	0.42	0.15	27	
Chemistry and mineralogy	39	0.54	0.07	47	2100 Natural sciences

Tabulka C.2: Analýza domén, datasetů a vlastností entit (2. část).

Doména	Počet entit	Horní mez	Spodní mez	Počet predikátů	Předpokládaná TaaS doména
Engineering and technology	45	0.5	0.1	37	1500 Industries and technology
Politics and government	47	0.22	0.08	49	0100 Politics and Administration
Religion, mysticism and mythology	47	0.2	0.06	38	2000 Social sciences
Warfare	47	0.54	0.08	48	
Education	48	0.51	0.15	96	2005 Social sciences
Health and medicine	53	0.53	0.13	37	2200 Medicine and pharmacy
Law	54	0.17	0.09	32	0200 Law
Culture and society	77	0.55	0.16	33	2005 Social sciences
Physics and astronomy	120	0.22	0.1	79	2100 Natural sciences
Biology	148	0.75	0.19	33	2100 Natural sciences
History	151	0.28	0.07	53	2005 Social sciences
Meteorology	154	0.89	0.12	36	
Art, architecture, and archaeology	166	0.4	0.1	25	2300 Arts
Transport	175	0.47	0.1	58	0100 Politics and Administration
Literature and theatre	185	0.37	0.12	25	2300 Arts
Video gaming	194	0.83	0.2	48	
Sport and recreation	208	0.54	0.15	29	2005 Social sciences
Geography and places	215	0.61	0.2	86	2100 Natural sciences
Music	215	0.69	0.25	39	2300 Arts
Media	290	0.47	0.16	52	2005 Social sciences
Vysvětlivky: Horní a spodní mez – Souvisí s predikáty z oddílu 2.3.1.					

Příloha D

Manuál aplikace na tvorbu trénovacích datasetů

D.1 Požadavky

- JDK 8
- Apache Maven
- GIT
- Stažené soubory DBpedia datasetu ve formátu .hdt a .hdt.index, například z `http://users.restdesc.org/rgverbor/tmp/hdt/`

D.2 Konfigurace

V souboru „src/main/resources/config.properties“ je nutné nastavit cestu souboru DBpedia.hdt. Ostatní možnosti v tomto souboru nemusí být měněny. Je nutné si dát pozor na zadání správných názvů složek a souborů.

D.3 Argumenty

Aplikace Tool zpracovává dva celočíselné argumenty:

- První argument určuje jaké entity jsou použity:
  - 1 – Mnou nalezené entity pro tři základní domény.
  - 2 – Featured Articles domény a jejich entity.
  - 3 – Pomocná metoda pro analýzu predikátů a jejich procentuálního zastoupení v entitách Featured Articles (druhý argument může zůstat 0)

- Druhý argument určuje jaké vlastnosti entit budou zpracovávány:
  - 1 – Jen predikáty (bez zálohy*).
  - 2 – Jen *rdf:type* (bez zálohy*).
  - 3 – Jen *dct:subject* (bez zálohy*).
  - 4 – Všechny vlastnosti entit:
    - Pro mnou nalezené entity se provede záloha* a výpis ARFF souborů.
    - Pro entity Featured Articles se provede jen záloha*.
  - 5 – Platí jen pro entity Featured Articles a provede obnovu ze zálohy* a výpis ARFF souborů.
  - 6 – Platí jen pro entity Featured Articles a provede to samé jako v předchozím bodě (5). Navíc provede rozšíření domén s malým počtem entit.

Doporučená konfigurace argumentů je „5 2“.

D.4 Vysvětlivky

Záloha* – Pomocná funkce, která uloží objekty Domain a Entity do JSON souborů k pozdějšímu použití.

D.5 Instalace a spuštění

Instalace a spuštění aplikace Tool probíhá z příkazové řádky systému Linux.

```
$ mkdir LearningDomainsForNamedEntities
$ cd LearningDomainsForNamedEntities/
$ git init
$ git pull https://benaktom@bitbucket.org/benaktom/
learningdomainsfornamedentities.git
$ cd LearningDomainsForNamedEntitiesTool/
$ mvn clean
$ mvn package
$ mvn exec:java -Dexec.args="2 5"
```

Příloha E

Manuál REST API

E.1 Požadavky

- JDK 8
- Apache Maven
- GIT
- Stažené soubory DBpedia datasetu ve formátu .hdt a .hdt.index, například z `http://users.restdesc.org/rgverbor/tmp/hdt/`

E.2 Konfigurace

V souboru „src/main/resources/config.properties“ je nutné nastavit cestu souboru DBpedia.hdt. Dále je možné nastavit model na kterém bude provedena predikce.

E.3 Instalace a spuštění

Instalace a spuštění REST API probíhá z příkazové řádky systému Linux. Pojmem spuštění je myšleno nastartování webového serveru *Embedded Glassfish*.

```
$ mkdir LearningDomainsForNamedEntities
$ cd LearningDomainsForNamedEntities/
$ git init
$ git pull https://benaktom@bitbucket.org/benaktom/
learningdomainsfornamedentities.git
$ cd LearningDomainsForNamedEntitiesApi/
$ mvn clean
$ mvn package
$ mvn mvn embedded-glassfish:run
```

E.4 Použití

Popis REST služby:

- Host: http://localhost:8181/LDFNE
- Resource: /entities/your_dbpedia_entity/domains
- Accept: application/json, text/plain

Příklad:

```
$ curl -H "Accept: application/json"
http://localhost:8181/LDFNE/entities/Berlin/domains
```

Příloha F

Experiment: algoritmy klasifikace

Tabulka F.1: Experiment: algoritmy klasifikace. Algoritmus Naive Bayes.

Algoritmus	Precision	Recall	F-measure
Naive Bayes	0.65	0.57	0.56

Tabulka F.2: Experiment: algoritmy klasifikace. Algoritmus SMO.

Algoritmus	Parametr C	Precision	Recall	F-measure
SMO	0.01	0.71	0.58	0.59
SMO	0.1	0.77	0.69	0.70
SMO	1.0	0.75	0.69	0.69
SMO	10.0	0.71	0.66	0.66
SMO	100.0	0.69	0.66	0.65

Tabulka F.3: Experiment: algoritmy klasifikace. Algoritmus kNN.

Algoritmus	Parametry			Precision	Recall	F-measure
	k	Vzdál. funkce	Váha			
kNN	1	EuclDist	no	0.72	0.66	0.66
kNN	1	EuclDist	1/d	0.72	0.66	0.66
kNN	1	EuclDist	1-d	0.72	0.66	0.66
kNN	2	EuclDist	no	0.72	0.64	0.65
kNN	2	EuclDist	1/d	0.72	0.65	0.66
kNN	2	EuclDist	1-d	0.72	0.65	0.66
kNN	3	EuclDist	no	0.71	0.63	0.64
kNN	3	EuclDist	1/d	0.71	0.65	0.65
kNN	3	EuclDist	1-d	0.71	0.64	0.65
kNN	1	ManhDist	no	0.72	0.66	0.66
kNN	1	ManhDist	1/d	0.72	0.66	0.66
kNN	1	ManhDist	1-d	0.72	0.66	0.66
kNN	2	ManhDist	no	0.72	0.64	0.65
kNN	2	ManhDist	1/d	0.72	0.66	0.66
kNN	2	ManhDist	1-d	0.72	0.65	0.66
kNN	3	ManhDist	no	0.71	0.63	0.64
kNN	3	ManhDist	1/d	0.71	0.65	0.65
kNN	3	ManhDist	1-d	0.71	0.64	0.65

Tabulka F.4: Experiment: algoritmy klasifikace. Algoritmus Random Forest (1. část).

Algoritmus	Parametry			Precision	Recall	F-measure
	Stromy	Hloubka	Vlastnosti			
Random Forest	50	inf	auto	0.75	0.71	0.70
Random Forest	100	inf	auto	0.75	0.71	0.71
Random Forest	150	inf	auto	0.75	0.71	0.71
Random Forest	50	10	auto	0.66	0.56	0.54
Random Forest	100	10	auto	0.67	0.57	0.55
Random Forest	150	10	auto	0.67	0.57	0.55
Random Forest	50	20	auto	0.74	0.65	0.65
Random Forest	100	20	auto	0.76	0.65	0.66
Random Forest	150	20	auto	0.76	0.66	0.66
Random Forest	50	30	auto	0.75	0.67	0.67
Random Forest	100	30	auto	0.76	0.68	0.69
Random Forest	150	30	auto	0.76	0.68	0.69
Random Forest	50	40	auto	0.75	0.69	0.69
Random Forest	100	40	auto	0.76	0.70	0.70
Random Forest	150	40	auto	0.76	0.70	0.70

Tabulka F.5: Experiment: algoritmy klasifikace. Algoritmus Random Forest (2. část).

Algoritmus	Parametry			Precision	Recall	F-measure
	Stromy	Hloubka	Vlastnosti			
Random Forest	50	inf	10	0.74	0.71	0.70
Random Forest	100	inf	10	0.75	0.71	0.70
Random Forest	150	inf	10	0.75	0.71	0.71
Random Forest	50	inf	20	0.74	0.70	0.70
Random Forest	100	inf	20	0.74	0.70	0.70
Random Forest	150	inf	20	0.74	0.70	0.70
Random Forest	50	inf	30	0.74	0.70	0.70
Random Forest	100	inf	30	0.74	0.71	0.70
Random Forest	150	inf	30	0.74	0.71	0.70
Random Forest	50	inf	40	0.73	0.70	0.69
Random Forest	100	inf	40	0.74	0.70	0.70
Random Forest	150	inf	40	0.74	0.70	0.70
Random Forest	150	50	15	0.76	0.71	0.71
Random Forest	150	100	15	0.76	0.72	0.72
Random Forest	150	150	15	0.75	0.71	0.71
Random Forest	50	100	15	0.75	0.71	0.71
Random Forest	100	100	15	0.76	0.72	0.71

Tabulka F.6: Experiment: algoritmy klasifikace. Výpočetní časy algoritmů (1. část).

Algoritmus	**Training**	**Testing**
Naive Bayes	0.86	4.04
SMO C=0.01	11.65	1.13
SMO C=0.1	20.17	1.14
SMO C=1.0	25.84	1.12
SMO C=10.0	28.30	1.12
SMO C=100.0	31.85	1.14
kNN k=1, vzdál.f.=EuclDist, váha=no	0.00	8.26
kNN k=1, vzdál.f.=EuclDist, váha=1/d	0.00	8.92
kNN k=1, vzdál.f.=EuclDist, váha=1-d	0.00	9.27
kNN k=2, vzdál.f.=EuclDist, váha=no	0.00	12.20
kNN k=2, vzdál.f.=EuclDist, váha=1/d	0.00	12.66
kNN k=2, vzdál.f.=EuclDist, váha=1-d	0.00	12.43
kNN k=3, vzdál.f.=EuclDist, váha=no	0.00	14.41
kNN k=3, vzdál.f.=EuclDist, váha=1/d	0.00	14.10
kNN k=3, vzdál.f.=EuclDist, váha=1-d	0.00	13.80
kNN k=1, vzdál.f.=ManhDist, váha=no	0.00	8.23
kNN k=1, vzdál.f.=ManhDist, váha=1/d	0.00	8.19
kNN k=1, vzdál.f.=ManhDist, váha=1-d	0.00	7.98

Tabulka F.7: Experiment: algoritmy klasifikace. Výpočetní časy algoritmů (2. část).

Algoritmus	**Training**	**Testing**
kNN k=2, vzdál.f.=ManhDist, váha=no	0.00	10.45
kNN k=2, vzdál.f.=ManhDist, váha=1/d	0.00	10.10
kNN k=2, vzdál.f.=ManhDist, váha=1-d	0.00	10.28
kNN k=3, vzdál.f.=ManhDist, váha=no	0.00	11.31
kNN k=3, vzdál.f.=ManhDist, váha=1/d	0.00	11.62
kNN k=3, vzdál.f.=ManhDist, váha=1-d	0.00	11.77
Random Forest strom=50, hl.=inf, vlast.=auto	15.66	0.04
Random Forest strom=100, hl.=inf, vlast.=auto	31.20	0.08
Random Forest strom=150, hl.=inf, vlast.=auto	46.45	0.14
Random Forest strom=50, hl.=10, vlast.=auto	0.88	0.01
Random Forest strom=100, hl.=10, vlast.=auto	1.77	0.01
Random Forest strom=150, hl.=10, vlast.=auto	2.55	0.02
Random Forest strom=50, hl.=20, vlast.=auto	1.77	0.01
Random Forest strom=100, hl.=20, vlast.=auto	3.56	0.02
Random Forest strom=150, hl.=20, vlast.=auto	5.21	0.03
Random Forest strom=50, hl.=30, vlast.=auto	2.57	0.01
Random Forest strom=100, hl.=30, vlast.=auto	5.22	0.03
Random Forest strom=150, hl.=30, vlast.=auto	7.79	0.04
Random Forest strom=50, hl.=40, vlast.=auto	3.51	0.02
Random Forest strom=100, hl.=40, vlast.=auto	6.83	0.03
Random Forest strom=150, hl.=40, vlast.=auto	10.23	0.05
Random Forest strom=50, hl.=inf, vlast.=10	15.53	0.04
Random Forest strom=100, hl.=inf, vlast.=10	31.51	0.10
Random Forest strom=150, hl.=inf, vlast.=10	45.94	0.15
Random Forest strom=50, hl.=inf, vlast.=20	16.53	0.03
Random Forest strom=100, hl.=inf, vlast.=20	32.87	0.07
Random Forest strom=150, hl.=inf, vlast.=20	49.82	0.11
Random Forest strom=50, hl.=inf, vlast.=30	17.13	0.03
Random Forest strom=100, hl.=inf, vlast.=30	35.83	0.06
Random Forest strom=150, hl.=inf, vlast.=30	53.77	0.10

Tabulka F.8: Experiment: algoritmy klasifikace. Výpočetní časy algoritmů (3. část).

Algoritmus	**Training**	**Testing**
Random Forest strom=50, hl.=inf, vlast.=40	18.24	0.02
Random Forest strom=100, hl.=inf, vlast.=40	36.87	0.05
Random Forest strom=150, hl.=inf, vlast.=40	54.01	0.09
Random Forest strom=150, hl.=50, vlast.=15	14.68	0.07
Random Forest strom=150, hl.=100, vlast.=15	30.60	0.12
Random Forest strom=150, hl.=150, vlast.=15	41.73	0.13
Random Forest strom=50, hl.=100, vlast.=15	10.59	0.03
Random Forest strom=100, hl.=100, vlast.=15	22.91	0.08

Vysvětlivky:
Training – User CPU time training – Výpočetní čas trénovaní v sekundách.
Testing – User CPU time testing – Výpočetní čas testování v sekundách.
vzdál.f. – Vzdálenostní funkce.
EuclDist – *EuclideanDistance*.
ManhDist – *ManhattanDistance*.
strom – Počet stromů.
hl. – Maximální hloubka stromu.
vlast. – Počet vlastností (atributů).