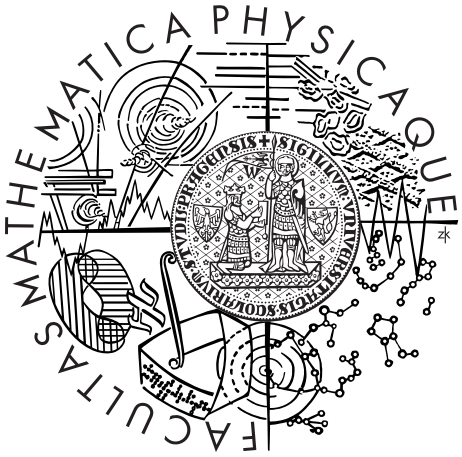
Univerzita Karlova v Praze
Matematicko-fyzikální fakulta

DIPLOMOVÁ PRÁCE



Bc. Vincent Kríž

Klasifikátor pro sémantické vzory užívání anglických sloves

Ústav formální a aplikované lingvistiky

Vedoucí diplomové práce:	RNDr. Martin Holub Ph.D.
Studijní program:	Informatika
Studijní obor:	Matematická lingvistika

Praha 2012

Rád by som poďakoval vedúcemu diplomovej práce za čas a cenné rady, ktorými výrazne prispel ku kvalite tejto práce.

Rád by som poďakoval mojim rodičom, ktorí ma podporovali nie len pri štúdiu, ale počas celého môjho života.

Rád by som poďakoval Karlovi Duškovi za podporu a trpezlivosť nie len pri písaní tejto práce.

Prehlasujem, že som túto diplomovú prácu vypracoval samostatne a výhradne s použitím citovaných prameňov, literatúry a ďalších odborných zdrojov.

Beriem na vedomie, že sa na moju prácu vzťahujú práva a povinnosti vyplývajúce zo zákona č. 121/2000 Zb., autorského zákona v platnom znení, predovšetkým skutočnosť, že Univerzita Karlova v Prahe má právo na uzavretie licenčnej zmluvy o použití tejto práce ako školského diela podľa §60 odst. 1 autorského zákona.

V Prahe dňa 5.4.2012 Podpis autora

Názov práce: Klasifikátor pro sémantické vzory užívání anglických sloves

Autor: Bc. Vincent Kríž

Katedra: Ústav formální a aplikované lingvistiky

Vedúci diplomovej práce: RNDr. Martin Holub Ph.D., Ústav formální a aplikované lingvistiky

Abstrakt: Cieľom tejto diplomovej práce je navrhnúť, implementovať a empiricky evaluovať klasifikátory pre rozpoznávanie sémantických patternov anglických slovies. Ako trénovacie a testovacie údaje používame konkordancie z pilotnej kolekcie 30 anglických slovies, ktorá bola spracovaná metódou Corpus Pattern Analysis. Modely klasifikátorov tvoríme pomocou algoritmov strojového učenia s učiteľom. Experimentujeme s rozhodovacími stromami, algoritmom k najbližších susedov (kNN), podpornými vektormi (SVM) a Adaboostom. V práci sa, okrem iného, zameriavame na návrh vhodnej množiny rysov pre strojové učenie (*feature selection*). Experimentujeme s množinami morfo-syntaktických i sémantických rysov. Naše výsledky ukazujú, že morfo-syntaktické rysy sú najdôležitejšie pre sémantickú desambiguáciu, hoci pre *niektoré* slovesá hrajú sémantické rysy dôležitú úlohu.

Kľúčové slová: lexikálna sémantika, anglické slovesá, strojové učenie, automatická klasifikácia, Corpus Pattern Analysis, Word Sense Disambiguation

Title: Classifier for semantic patterns of English verbs

Author: Bc. Vincent Kríž

Department: Institute of formal and applicational lingvistics

Supervisor: RNDr. Martin Holub Ph.D., Institute of formal and applicational lingvistics

Abstract: The goal of the diploma thesis is to design, implement and evaluate classifiers for automatic classification of semantic patterns of English verbs according to a pattern lexicon that draws on the Corpus Pattern Analysis. We use a pilot collection of 30 sample English verbs as training and test data sets. We employ standard methods of machine learning. In our experiments we use decision trees, k-nearest neighbourghs (kNN), support vector machines (SVM) and Adaboost algorithms. Among other things we concentrate on feature design and selection. We experiment with both morpho-syntactic and semantic features. Our results show that the morpho-syntactic features are the most important for statistically-driven semantic disambiguation. Nevertheless, for *some* verbs the use of semantic features plays an important role.

Keywords: lexical semantics, English verbs, machine learning, automatic classification, Corpus Pattern Analysis, Word Sense Disambiguation

Obsah

Kapitola 1

Úvod

Ako sa uvádza v [1] a tiež v [2], lexikálna desambiguácia je klasickým otvoreným problémom v oblasti počítačovej lingvistiky. Problém bol formulovaný v samých počiatkoch počítačového spracovania prirodzeného jazyka. Už Warren Weaver vo svojom memorande o strojovom preklade [3] naznačil potrebu rozlišovania kontextu. Tradičným prístupom k riešeniu problému lexikálnej desambiguácie je klasická definícia úlohy *Word-Sense Disambiguation* (WSD) [2, 36].

Ako uvádzajú autori [4], hlavným cieľom WSD je identifikovať zmysel (význam) slova použitého v konkrétnej vete, keďže slová majú spravidla niekoľko významov. Určovanie zmyslu prebieha najčastejšie tzv. sémantickým značkovaním. To sa používa buď na rozpoznávanie sémanticky dôležitých objektov, alebo na spájanie výskytov slov v korpusoch s najvhodnejšou sémantickou definíciou uvedenou v slovníku. Tento proces môže byť automatický, alebo manuálny.

V našej práci sa sústreďujeme na automatické priradenie sémantickej kategórie cieľovému slovu v zadanom kontexte. V práci sa zameriavame na anglické slovesá. Boli sme inšpirovaní metódou CPA (*Corpus Pattern Analysis*) [6] a jej implementáciou – slovníkom PDEV (*Pattern Dictionary of English Verbs*) [7, 8, 9].

CPA je pomerne nová metóda, ktorá sa usiluje o syntagmatické a sémantické popísanie (predovšetkým) anglických slovies. Jedná sa o dôsledne korpusovú, empirickú metódu, ktorá analyzuje typické vzory používania slov v korpuse a popisuje význam slovies pomocou kontextových preferencií definovaných syntakticky a sémanticky [8]. Slovník PDEV je sémantická konkordancia, ktorá je postavená na odlišných princípoch, ako známe projekty FrameNet [10], WordNet [11], PropBank [12], alebo OntoNotes [13].

Manuálne extrahované patterny[1] najčastejších použití slovies popisujú, jednoducho povedané, podobné udalosti, v ktorých vystupujú podobní účastníci (napríklad ľudia, inštitúcie, dopravné prostriedky). V porovnaní s inými sémantickými konkordanciami má slovník PDEV relatívne vysokú granularitu.

Vyberanie patternov v skutočnosti neznamená desambiguáciu konkordancie, ale určenie, ktorý pattern je konkordancii najviac podobný, čo je ľahšia úloha než samotná WSD. Tento princíp sa nám zdá byť sľubný pre slovesá, ktoré predstavujú najväčší problém pre WSD.

Rozpoznávanie sémantických patternov je nový prístup k sémantickému značkovaniu. Podľa metódy CPA nemajú slová fixné významy. Namiesto nich môžeme v korpusoch identifikovať pravidelné vzory, ktoré aktivujú príslušné významové potenciály slovesa.

Diplomová práca má analyzovať a čo najlepšie využiť dostupné údaje o typických vzoroch používaní anglických slovies pre konštrukciu automatických klasifikátorov. Cie-

[1] V ďalšom texte budeme používať toto kalkové slovo prevzaté z angličtiny, pretože nám nie je známy žiadny vhodný slovenský ekvivalent.

ľom práce je navrhnúť, implementovať a empiricky evaluovať klasifikátory pre rozpoznávanie patternov. Okrem iného sa predpokladá rozpoznávanie lexikálnych jednotiek realizujúcich jednotlivé sémantické typy v PDEV, využitie automatického parsingu angličtiny a metód strojového učenia [2, 16].

V druhej kapitole diplomovej práce podrobne predstavujeme metódu CPA a slovník PDEV. Popisujeme štruktúru patternov a sémantických značiek. Na záver definujeme zadanie klasifikačnej úlohy, ktorú v práci riešime.

V tretej kapitole podrobne popisujeme metódy strojového učenia, pomocou ktorých budeme implementovať jednotlivé modely klasifikátorov.

Vo štvrtej kapitole predstavujeme tradičné prístupy k WSD a predstavujeme tiež podobné práce, ktoré za zaoberali automatickým sémantickým značkovaním.

V piatej kapitole sa podrobne zaoberáme vstupnými údajmi, ktoré máme k dispozícii. Predstavujeme zaujímavé štatistiky a pohľady na dostupnú kolekciu údajov. Popisujeme tiež technický formát vstupných údajov.

V šiestej kapitole sa zaoberáme návrhom rysov pre algoritmy strojového učenia. Predstavujeme množinu morfo-syntaktických rysov a tri množiny sémantických rysov, ktoré použijeme v algoritmoch strojového učenia.

V siedmej kapitole pripravujeme inštancie pre strojové učenie. Rozdeľujeme inštancie na trénovacie a testovacie, definujeme prahové frekvencie výstupných tried a na záver stanovujeme baseline pre naše experimenty.

V ôsmej kapitole popisujeme experimenty, ktorými sme trénovali a ladili modely klasifikátorov a predstavujeme bohatú sériu experimentov, ktorými sme sa snažili optimalizovať navrhnuté množiny rysov. Na záver kapitoly prezentujeme úspešnosti modelov na testovacích inštanciách.

Zhrnutie experimentov, závery a výhľady do budúcnosti prezentujeme v závere práce.

Kapitola 2

Pattern Dictionary of English Verbs (PDEV)

V tejto kapitole predstavujeme čitateľovi projekt PDEV ako prvú implementáciu metódy CPA. Popisujeme štruktúru slovníka a uvádzame prehľad experimentov, ktoré už so slovníkom PDEV prebehli a boli publikované v iných prácach.

2.1 Metóda CPA a slovník PDEV

Slovník PDEV [7] je vyvíjaný od roku 2004. Jeho hlavným rysom je praktické použitie novej metódy CPA [6].

Metóda CPA dôsledne dodržuje Sinclairov koncept zachytenia významov v typických vzoroch použitia jazyka [17]. John Sinclair, ktorý je považovaný za nestora korpusovej lingvistiky, vytvoril tento koncep ako opozíciu ku doterajšiemu kritizovanému lexikografickému postupu, ktorý oddeľoval lexikón a gramatiku. Gramatika v krajných prípadoch popisuje len formu lexikálnej jednotky v súvislosti s jej potencionálnym kontextom, zatiaľ čo lexikón popisuje význam, ktorý je obsiahnutý v základnom tvare lexikálnej jednotky a to bez ohľadu na jej kontext. Podľa Johna Sinclaira sú význam a forma úzko prepojené, dokonca ich možno považovať až za identické, pretože väčšinu ambiguít v jazyku je možné rozhodnúť práve na základe znalosti kontextu.

Metóda je založená na pozorovaní, že napriek tomu, že je veľa slov, ktoré majú vysokú mieru ambiguity, vzory použitia slov (patterny) sú ambiguitné len veľmi zriedka. CPA sa preto snaží

1. identifikovať patterny bežných použití slov;
2. asociovať významy slov s patternami, namiesto určovania významu izolovaných slov.

Stačí zbežný pohľad do korpusov a je okamžite vidieť, že väčšina použití slov je prekvapivo pravidelná a dá sa zaradiť do jedného z niekoľkých patternov. CPA sa nesnaží zachytiť všetky možné realizácie (použitia) slovesa v jazyku. Sústreďuje sa len na najčastejšie použitia pomocou relatívne malého počtu patternov. Tieto patterny následne prehlasuje za tzv. *normy*.

Prvou aplikáciou metódy CPA je práve slovník PDEV. Jeho uplatnenie vidíme v pomoci študentom pri štúdiu a učiteľom pri učení angličtiny a predovšetkým v počítačovom spracovaní prirodzeného jazyka ako nástroj desambiguácie významu slov.

Slovník PDEV je zbierkou použití vybraných anglických slovies. Ku každej korkondancii je ručne priradená značka patternu. Korkondancie, ktoré slovník PDEV obsahuje, sú vybrané z Britského národného korpusu.

Britský národný korpus (BNC, z anglického *British National Corpus*) je jedným z najznámejších a najvýznamnejších korpusov angličtiny v súčasnosti. Obsahuje približne 100 miliónov slovných tokenov a predstavuje tak významnú zbierku písaných a hovorených vzoriek z anglického jazyka. Texty v korpuse pochádzajú zo širokého spektra zdrojov a štýlov a boli zostavené tak, aby tvorili reprezentatívnu vzorku britskej angličtiny v druhej polovici 20. storočia. [18]

Britský národný korpus obsahuje okrem písaných textov aj ortografické prepisy neformálnych rozhovorov a niektoré špeciálne žánre, ktoré neboli vhodné pre projekt PDEV. Písané texty majú tendenciu byť zostavované pozornejšie. Týka sa to výberu slov a premysleného slovosledu. Bývajú plánované dopredu pred ich napísaním. Obsahujú tak menej chýb, váhaní, nedokončených viet a podobne. V slovníku PDEV je používaná len časť korpusu, ktorá obsahuje asi 50 miliónov slovných tokenov. Tento korpus budeme ďalej v texte označovať ako BNC50.

2.1.1 Štruktúra patternov

Okrem samotných inštancií sú nedeliteľnou súčasťou slovníku PDEV definície patternov.

Definícia patternu obsahuje niekoľko globálnych atribútov, ktoré priradzujú nejakú vlastnosť celému patternu. Hlavnou časťou definície je potom popis kolokačných pozícií (subjekt, objekty, adverbiály) a implikatúra. Popis kolokačných pozícií je uvádzaný v tzv. propozícii.

Príklad zápisu patternov pre sloveso *submit* je uvedený v tabuľke 2.1. V zápise sa používajú nasledujúce konvencie:

- Sémantické typy sú uvádzané v dvojitých hranatých zátvorkách, napr. [[**Human**]]
- Zložené zátvorky slúžia na zhlukovanie, napr. {approval | discussion | arbitration | inspection | designation | assessment | funding | taxation | ... }
- Nepovinné argumenty sú uvádzané v okrúhlych zátvorkách, napr. (**Self**)

V nasledujúcom texte popisujeme všetky časti patternov podrobne.

Globálne atribúty patternu

Ako sme už naznačili vyššie, globálne atribúty sa vzťahujú k celému patternu. Určujú napríklad, či sa jedná o idióm, frázové sloveso, či sloveso vyžaduje objekt a pod. Pomocou globálnych atribútov je tiež možné definovať doménu, v ktorej sa pattern používa.

Popis subjektu

Pozícia subjektu je popísaná pomocou tzv. sémantického typu, ktorý môže byť navyše upresnený pomocou sémantickej role. Napríklad v tabuľke 2.1, v patterne č. 1 je sémantický typ [[**Institution**]] upresnený sémantickou rolou **Competitor**.

V prípade, že sa sémantický typ vyskytuje v definícii patternu viac ako raz, je doplnený číslom, aby ho bolo možné identifikovať jednoznačne. Príkladom môže byť typ [[**Human 1**]] v definícii patternu č. 5 v tabuľke 2.1.

Ďalšou možnosťou, pomocou ktorej je možné definovať kolokačnú pozíciu, je použitie lexikálnej množiny. Obsahuje zoznam lexikálnych jednotiek, ktoré sa na danej pozícii vyskytujú typicky. Lexikálna množina môže byť uvedená samostatne, alebo ako doplnenie k sémantickému typu. Je potrebné zdôrazniť, že lexikálna množina nevyjadruje

Č.	Pattern / Implikatúra
1	[[**Human 1** | **Institution 1**] ^ [**Human 1** | **Institution 1 = Competitor**]] **submit** [[**Plan** | **Document** | **Speech Act** | **Proposition** | **{complaint** | **demand** | **request** | **claim** | **application** | **proposal** | **report** | **resignation** | **information** | **plea** | **petition** | **memorandum** | **budget** | **amendment** | **programme** | **...}**] ^ [**Artifact** | **Artwork** | **Service** | **Activity** | **{design** | **tender** | **bid** | **entry** | **dance** | **...}**]] (({**to**} **Human 2** | **Institution 2 = authority**)^({**to**} **Human 2** | **Institution 2 = referee**)) ({**for**} {**approval** | **discussion** | **arbitration** | **inspection** | **designation** | **assessment** | **funding** | **taxation** | **...**}) [[Human 1 | Institution 1]] presents [[Plan | Document]] to [[Human 2 | Institution 2]] for {approval | discussion | arbitration | inspection | designation | assessment | taxation | ...}
2	[**Human** | **Institution**] **submit** [**THAT-CL**|**QUOTE**] [[Human | Institution]] respectfully expresses {that [CLAUSE]} and invites listeners or readers to accept that {that [CLAUSE]} is true}
4	[**Human 1** | **Institution 1**] **submit (Self) ({to} Human 2** | **Institution 2)** [[Human 1 | Institution 1]] acknowledges the superior force of [[Human 2 | Institution 2]] and puts [[Self]] in the power of [[Human 2 | Institution 2]]
5	[**Human 1**] **submit (Self)** [[{**to**} **Eventuality = Unpleasant**] ^ [{**to**} **Rule**]] [[Human 1]] accepts [[Rule |Eventuality = Unpleasant]] without complaining
6	[**passive**] [**Human**| **Institution**] **submit** [**Anything**] [{**to**} **Eventuality**] [[Human 1|Institution 1]] exposes [[Anything]] to [[Eventuality]]

Tabuľka 2.1: Príklad definácií patternov pre sloveso *submit*.

všetky možné lexikálne jednotky, ktoré sa môžu na danej pozícii vyskytnúť, ale len najčastejšie alebo najtypickejšie. Príklad lexikálnej množiny môže čitateľ vidieť v definícii patternu č. 1 v tabuľke 2.1.

Definíciu subjektu je možné niekoľkokrát opakovať. V tomto prípade sa hovorí o tzv. alternácii subjektu. Je potrebné zdôrazniť, že alternácia subjektu neznamená, že vo vete očakávame niekoľko subjektov, ale práve jeden z definovaných.

Na záver popisu subjektu ešte uvádzame, že pod pojmom subjekt v slovníku PDEV máme na mysli vždy vykonávateľa deja (agenta).

Popis objektu

Objekt rozlišujeme priamy a nepriamy. Nepriamy objekt nie je príliš častý. Objekt môže byť definovaný pomocou rovnakých prostriedkov ako subjekt (tj. pomocou sémantických typov, sémantických rolí a lexikálnych množín).

V definícii patternu sa môže vyskytovať niekoľko definícií objektov.

Popis adverbiálov

Ako adverbiál je v slovníku PDEV označovaná predložková fráza alebo príslovkové určenie.

Adverbiálov môže byť v jednom patterne definovaných niekoľko, pričom každá definícia môže mať niekoľko alternácií. Definícia adverbiálu sa skladá z definície predložkového subjektu (môže byť definovaný pomocou sémantických typov, rolí a lexikálnych množín) alebo príslovkového určenia a ďalších atribútov, medzi ktoré patrí napríklad funkcia adverbiálu, jeho obligatórnosť a predložka.

Popis komplementu

Komplement môže byť subjektový alebo objektový. V slovníku PDEV sa komplement nevyskytuje príliš často.

Popis slovesných klauzúl

Niektoré argumenty PDEV rozlišuje na základe ich povrchovej štruktúry. Môže sa jednať o tieto klauzule:

- that-clause
- wh-clause
- to+inf
- -ing
- quote

Implikatúra

Vysvetľuje význam daného patternu pomocou rovnakých sémantických typov, ktoré sú uvedené v popise kolokačných pozícií.

2.1.2 Značky pre anotáciu

Štatistické analýzy korpusov viedli k poznaniu, že pre každé sloveso existuje malé množstvo patternov, do ktorých spadá väčšina inštancií. Tieto patterny označujeme ako tzv. *normy*. Existujú ale aj inštancie, o ktorých môžeme povedať, že sú z daných noriem derivované. Môžu mať rovnaký význam ako normy, ale použitie slovesa napriek tomu vykazuje určitú odchýlku (sémantickú alebo syntaktickú), alebo, naopak, môžu mať iný význam a rovnakú syntaktickú štruktúru ako norma. Označujeme ich ako tzv. *exploatácie*.

Patterny, ktoré považujeme za normy sú v slovníku PDEV rozlišované číslami. Každá inštancia, ktorú anotátor priradil k určitému patternu je anotovaná číslom svojho patternu. Ak bola inštancia vyhodnotená ako exploatácia, za číslo patternu sa navyše priradí jedna zo značiek `.a`, `.c`, `.f` alebo `.s`, podľa toho, o akú exploatáciu sa jedná.

V slovníku PDEV rozlišujeme 4 druhy exploatácií. Príklady uvedené v definíciách sme prevzali z Anotačného manuálu pre anotátorov projektu PDEV [9]. Okrem 4 druhov exploatácií obsahuje značková sada (*tagset*) slovníka PDEV ešte ďalšie 2 značky určené pre špeciálne prípady. V ďalšom texte podrobne popisujeme jednotlivé druhy značiek.

Anomálny argument (`.a`)

Značka `.a` indikuje *anomálny argument* alebo *čestného člena* lexikálnej množiny. Čestný člen množiny je podstatné meno, ktoré je nezvyčajné alebo neočakávané a nemá sémantický význam popísaný sémantickým typom i napriek tomu, že význam použitia súhlasí s implikatúrou patternu. Príkladom je inštancia

> *Rashid Solh, the prime minister, spoke darkly of the arrival in Lebanon of several hundred Israeli agents provocateurs whose mission was to destroy the republic. The plot had* ***arrived*** *at Beirut.*

Význam druhej vety je jasný a skoro úplne zodpovedá patternu

[[**Human** | **Vehicle** | **Animal**]] arrive [NO OBJ] {at [[**Location**]]}

Výraz the plot však nepatrí ani do jedného zo sémantických typov definovaných pre subjekt. Naopak, je veľmi nezvyčajné, že *plot* ***arriving*** *at a place*. Na základe tejto úvahy bude inštancia označená ako exploatácia s anomálnym argumentom.

Coercion (.c)

Coercion je špeciálny druh anomálneho argumentu. V týchto prípadoch je tak jasné, aké slovo je očakávané na danej pozícii vo vete, že ho ľudia ani nemajú potrebu vyslovovať. Napríklad, všetci predsa vedia, že sa pije len tekutina. Namiesto takéhoto slova potom človek použije slovo, ktoré prináša ešte viac informácie. V prípade príkladu s tekutinou to bude nejaký kontajner. V jednej vete tak získame navyše informáciu o tom, koľko nápoja bolo vypitého. Príkladom je inštancia

She ***drank*** *8 glasses of spirits.*

Táto inštancia je označená ako coercion patternu

[[**Human**]] drink [[**Beverage**]]

Figuratívne použitie (.f)

Príkladom metaforického použitia slovesa arrive môže byť inštancia

Jane and I quickly ***arrived*** *at joint decisions about the project.*

Napriek tomu, inštancia sa presne zhoduje s patternom

[[**Human** | **Institution**]] arrive [NO OBJ] {at [[**Concept** = **Considered Opinion**]]}

V takýchto prípadoch je inštancia ohodnotená ako figuratívne použitie patternu.

Nezvyčajná syntax (.s)

Uvažujme nasledujúcu inštanciu:

We ***punish*** *too much—and in particular, we* ***imprison*** *too much.*

Ako objekt oboch slovies môžeme uvažovať ľudí, alebo niekoho, kto porušuje zákony. Jedná sa o syntaktickú exploatáciu. Slovesá sa používajú ako intranzitívne, aj keď v bežnom používaní vyžadujú objekty.

Neklasifikovateľné (u)

Ako sme už spomenuli v úvodnej časti venovanej metóde CPA, slovník PDEV sa snaží zachytiť len najčastejšie formy použitia slovies, ktoré nazýva normami. V slovníku sa však vyskytujú aj inštancie, ktoré nie sú zaraditeľné do žiadneho z definovaných patternov. Aj tieto inštancie museli anotátori označiť značkou a preto bola definovaná značka u. Sú ňou označené inštancie, ktoré nie sú zaraditeľné do niektorého z existujúcich patternov.

Pri tvorbe slovníka sa niekoľkokrát stalo, že pridaním nových dát narástlo množstvo inštancií so značkou u natoľko, že v tejto skupine bolo možné identifikovať nový pattern, ktorý sa zaradil medzi normálne použitia, pretože sa stal dostatočne frekventovaným.

Chyby taggera a ďalší šum (`x`)

Pri automatickom spracovaní korpusu a získavaní inštancií sa môže stať, že automatická procedúra zaradí medzi inštancie aj vety, v ktorých sa cieľové sloveso nenachádza, prípadne sa jedná o iný slovný druh. Často za to môže chybovosť taggeru. V týchto prípadoch anotátori používajú značku `x`, ktorá označuje inštancie, kde nie je môžné určiť pattern, pretože tam chýba cieľové sloveso.

Príkladom môže byť inštancia, ktorá sa nachádza v sade inštancií pre sloveso ally. V tomto prípade sa jedná o vlastné meno osoby, ktoré bolo označené ako sloveso:

Ally *McCoist salutes Rangers' second goal*

2.2 Kolekcia VPS-30-En

Údaje, ktoré v tejto práci využívame boli publikované v [19] a tvoria kolekciu VPS-30-En (*Verb Pattern Sample, 30 English verbs*). Túto kolekciu budeme v ďalšom texte označovať skratkou **VPS**. Jedná sa o pilotnú verziu nového lexikálneho zdroja, v ktorom sú slovesné lexikálne jednotky obohatené o sémanticky anotované inštancie z korpusu.

Pilotná kolekcia obsahuje týchto 30 anglických slovies: *access, ally, arrive, breathe, claim, cool, crush, cry, deny, enlarge, enlist, forge, furnish, hail, halt, part, plough, plug, pour, say, smash, smell, steer, submit, swell, tell, throw, trouble, wake* a *yield*.

Kolekcia VPS vznikla na základe slovníka PDEV. Originálne údaje zo slovníka PDEV boli prečistené a na základe podrobne vypracovaného anotačného manuálu [9] prebehlo niekoľko kôl, v ktorých vybrané vzorky inštancií anotovalo niekoľko nezávislých anotátorov.

Kolekcia VPS sa v niekoľkých aspektoch odlišuje od koncepcie slovníka PDEV. Kým slovník PDEV má svojím rozsahom tendenciu stať sa veľkým lexikálnym zdrojom, cieľom VPS je vytvoriť čo najčistejšiu vzorku použitia metódy CPA, ktorá by mohla byť použiteľná pri strojových experimetoch so sémantickým zhlukovaním založeným na metóde CPA.

2.3 Anotačné experimenty so slovníkom PDEV

Motiváciou k experimentom bol fakt, že nutným predpokladom použitia slovníka PDEV v aplikáciách strojového spracovania prirodzeného jazyka, napríklad pri strojovom preklade, je ľudská zhoda pri priradzovaní patternov jednotlivým konkordanciám. Vznikla preto potreba overiť, či má zmysel pokúsiť sa určovať rozpoznávanie patternov strojovo. Dobrým dôkazom by bola práve priateľná medzianotátorská zhoda viacerých anotátorov pri priradzovaní patternov konkrétnym konkordanciám.

2.3.1 Testovacie anotácie originálneho PDEV

Prvé experimenty s anotovaním korkondancií slovníka PDEV viacerými anotátormi boli popísané v [14]. Cieľom experimentu bolo predovšetkým meranie medzianotátorskej zhody.

Zo slovníka PDEV bolo vybraných 30 slovies, ktoré tvorili reprezentatívnu vzorku – čo do pokrytia slovesa v korpuse a tiež čo do počtu patternov, ktoré mali jednotlivé slovesá. Pre každé sloveso sa náhodne vybralo 50 viet z korpusu BNC50 alebo PEDT. Tieto vety potom nezávisle od seba anotovali dvaja, traja, alebo štyria anotátori. V čase experimentu sa ešte nerozlišovali exploatácie `.a`, `.c`, `.f` a `.s`. Namiesto nich mohli anotátori použiť jedinú značku `.e`. Anotátori mohli tiež používať značky `x` a `u`.

Na meranie medzianotátorskej zhody medzi 2 anotátormi bola použitá miera Cohenova kapa [20]. V prípade vyššieho počtu anotátorov bola použitá miera Fleissova kapa [21].

Z výsledkov experimentu vyplynulo niekoľko dôležitých poznatkov:

1. V teoretických základoch k metóde CPA nie je určené, aký široký kontext sa má uvažovať pri priradzovaní patternov jednotlivým vetám.

2. V niektorých prípadoch (napr. participium) bolo ťažké rozhodnúť, či slovo ešte považovať za sloveso alebo už nie.

3. Problém pri priradzovaní patternov sú elipsy. Vypustením argumentu môže dôjsť k tomu, že veta vyhovuje dvom alebo viacerým patternom.

4. V ojedinelých prípadoch sa stalo, že argument slovesa bolo možné zaradiť do rôznych sémentických typov a tým celú vetu klasifikovať dvoma rôznymi patternami.

5. Anotátori mali v niektorých prípadoch problém s angličtinou a zle porozumeli niektorým vetám.

6. Významná nezhoda medzi anotátormi sa prejavila v prípade, že v PDEV neexistoval vhodný pattern pre anotované vety. Anotátori buď vybrali existujúci pattern, ktorý nevyhovoval definícii úplne (a pridali k patternu značku `.e`) alebo zvolili značku `u`.

7. Objavili sa patterny, ktoré boli definované príliš *jemne* a anotátori si medzi nimi nevedeli vybrať.

8. Objavili sa sémantické typy, ktoré boli definované príliš *jemne*. Takéto typy použité v definíciách patternov následne znemožňovali použitie patternov aj pre vety, ktoré intuitívne k týmto patternom patrili.

Napriek týmto nedostatkom, ktoré zhoršujú výslednú medzianotátorskú zhodu, je pozitívne, že nezhody pri samotnom priradzovaní patternov neboli medzi anotátormi príliš časté. Nezanedbateľný počet nezhôd vznikol aj kvôli nepozornosti anotátorov, ktorí omylom priradzovali k vetám nesprávne číslo patternu, aj keď mali na mysli správny pattern, prípadne si nevšimli, že existuje ešte presnejší pattern, ktorý sa viac hodí na zadanú vetu.

2.3.2 Anotácia kolekcie VPS

Kolekcia VPS bola anotovaná prepracovanou metodológiou prezentovanou v [19]. V prvej fáze lexikograf revidoval referenčnú vzorku konkordancií anotovanú P. Hanksom, tak ako sa vyskytovala v slovníku PDEV. Všetky exploatácie označené ako `.e` boli preklasifikované podľa novej schémy popísanej v časti 2.1.2. Ak to bolo vhodné, lexikograf revidoval i samotné definície patternov. Vo väčšine prípadov boli len pridané nové sémantické typy a prvky lexikálnych množín. V niektorých prípadoch boli dva patterny zlúčené do jedného, prípade jeden pattern rozdelený na dva, aby sa dosiahlo optimálne priradzovanie konkordancií k patternom.

V druhej fáze dostali anotátori 50 náhodne vybraných konkordancií spolu s referenčnou vzorkou. Anotátori zaradili každú konkordanciu k jednému patternu. Po ukončení anotovania bola zmeraná medzianotátorská zhoda (IAA, *Inter-Annotator Agreement*) a všetky nezhody boli manuálne analyzované. Analýza bola podporená maticami konfúzie, ktoré boli spočítané pre každú dvojicu anotátorov.

V prípade, že miera IAA dosiahla uspokojivé hodnoty, všetky nezhody boli manuálne adjudikované lexikografom, ktorý vytvoril zoznam všetkých akceptovaných anotácií a jednu značku prehlásil za *najlepšiu voľbu*.

Miera IAA spolu s maticami konfúzie slúžila aj ako spätná väzba lexikografovi, ktorý na základe výsledkov anotácie mohol prehodnotiť vhodnosť definície niektorých patternov. Po zmene definícií sa zmenili tiež adjudikované značky a aktualizované anotované množiny konkordancií sa pripojili k referenčnej vzorke.

Výsledkom anotácií bolo vytvorenie zlatého štandardu (*gold standard data set*). Z pravidla obsahuje 350 konkordancií (250 konkordancií z referenčnej vzorky, jednu skúšobne adjudikovanú vzorku a finálnu adjudikovanú vzorku).

2.3.3 Meranie medzianotátorskej zhody

Na vyjadrenie miery medzianotátorskej zhody medzi viac ako dvomi anotátormi sa používa miera Fleissova kapa [21]. Jedná sa o štatistickú mieru zhody medzi určitým, pevne daným počtom anotátorov, ktorí anotujú rovnaké inštancie pomocou danej množiny značiek. Podobnou mierou, ktorá sa často na meranie medzianotátorskej zhody používa, je Cohenova kapa [20]. Tá je však definovaná len na meranie zhody medzi dvoma anotátormi.

Formálne môžeme Fleissovu kapu definovať nasledujúcim spôsobom: Nech N je počet multianotovaných inštancií, ktoré sú indexované číslom $i = 1, ..., N$. Nech n je počet anotátorov a k počet možných značiek, ktoré majú anotátori k dizpozícii a ktoré sú indexované číslom $j = 1, ..., k$. Označme číslom n_{ij} počet anotátorov, ktorí i-tu inštanciu anotovali značkou j.

Definujeme pravdepodobnosť p_j, ako pravdepodobnosť, že bola použitá značka j:

$$p_j = \frac{1}{Nn} \sum_{i=1}^{N} n_{ij}$$

Definujeme číslo P_i, ktoré vyjadruje podiel medzi počtom zhôd medzi dvoma anotátormi na inštancii i a celkovým počtom zhôd, ktoré mohli nastať medzi každými dvoma anotátormi na inštancii i:

$$P_i = \frac{1}{n(n-1)} \sum_{j=1}^{k} n_{ij}(n_{ij} - 1)$$

Následne môžme definovať Fleissovu kapu κ nasledujúcim predpisom:

$$\kappa = \frac{\bar{P} - \bar{P}_e}{1 - \bar{P}_e},$$

pričom platí, že

$$\bar{P} = \frac{1}{N} \sum_{i=1}^{N} P_i \quad a \quad \bar{P}_e = \sum_{j=1}^{k} np_j^2$$

Počas práce na kolekcii VPS sa zistilo [4], že Fleissova kapa nie je spoľahlivou mierou na určenie zhody medzi anotátormi. Jej zásadným problémom je, že nezohľadňuje mieru sémantickej granularity. Tú si jednoducho môžeme predstaviť ako počet patternov pre jednotlivé slovesá. Ak by sme zvolili veľkú granularitu, znamenalo by to, že rozlišujeme veľa významov (a definovali by sme veľké množstvo patternov). Sú známe experimenty, ktoré dokazujú, že ak sa anotátorom predložia priveľmi granulované významy, nastane veľká nezhoda a Fleissova kapa bude malá. Naopak, príliš malá sémantická granularita by znamenala len zopár značiek a medzi anotátormi by nastala veľká zhoda. Znižovanie

sémantickej granularity má však za následok, že strácame podstatnú informáciu – tj. rozlišovanie medzi významami na úkor vyššej medzianotátorskej zhody.

V prácach [4] a [5] je popísaná nová miera pre vyjadrenie IAA. Nazýva sa ARG (*Avarage Reliable Gain*). Počíta sa na základe matíc konfúzie. Tabuľka 2.3.3 porovnáva výsledky merania medzianotátorskej zhody pomocou Fleissovej kapy a pomocou miery ARG.

Sloveso	**Fleiss**	**ARG**	**Sloveso**	**Fleiss**	**ARG**
access	0.683	0.762	part	0.864	2.068
ally	0.733	1.229	plough	0.953	2.803
arrive	0.917	1.421	plug	0.722	1.336
breathe	0.835	1.309	pour	0.745	1.601
claim	0.840	1.303	say	0.876	0.801
cool	0.875	1.846	smash	0.787	1.267
crush	0.634	0.748	smell	0.882	1.716
cry	0.826	1.221	steer	0.722	1.195
deny	0.691	1.145	submit	0.879	0.879
enlarge	0.623	0.646	swell	0.804	1.518
enlist	0.859	1.417	tell	0.915	1.381
forge	0.634	1.069	throw	0.613	0.721
furnish	0.798	1.251	trouble	0.763	1.525
hail	0.835	0.943	wake	0.773	1.121
halt	0.697	0.521	yield	0.755	1.107

Tabuľka 2.2: Miera IAA meraná pomocou miery Fleissova kapy a pomocou novej miery ARG.

2.4 Definícia klasifikačnej úlohy

Kým ľudskí používatelia slovníka PDEV môžu intuitívne metódu CPA označiť za prínosnú, nemáme doteraz žiaden dôkaz, že je táto metóda vhodná aj pre strojové spracovanie prirodzeného jazyka.

Prvým predpokladom pre použitie slovníka pri storojovom spracovaní jazyka je, že na klasifikácii použití slov do jednotlivých patternov sa ľudia (anotátori) zhodujú a zaradzujú použitia do rovnakých patternov.

Samotným dôkazom použiteľnosti tejto metódy je následne úloha riešená v tejto práci – vytvorenie automatických klasifikátorov, ktoré by s dostatočnou úspešnosťou zaradzovali použitia slov do jednotlivých patternov.

Cieľom tejto diplomovej práce je navrhnúť, implementovať a empiricky evaluovať klasifikátory pre rozpoznávanie patternov anglických slovies.

Úloha bude riešená pomocou metód strojového učenia s učiteľom. Kontrétne bude použitý algoritmus Rozhodovacích stromov, algoritmus K najbližších susedov, algoritmus Podporných vektorov a algoritmus Adaboost. Metódy strojového učenia použijú na trénovavie a testovanie inštancie z kolekcie VPS.

Podstatnou časťou práce je navrhnutie a optimalizácia rysov vhodných pre algoritmy strojového učenia. Predpokladá sa použitie morfo-syntaktických informácií získaných z nástrojov morfologickej analýzy (tagger), syntaktickej analýzy (parser) a rozpoznávača menných entít (NER). Cieľom práce je tiež vytvoriť vhodnú množinu sémantických rysov, ktoré budú slúžiť na rozpoznávanie lexikálnych jednotiek realizujúcich jednotlivé sémantické typy v PDEV.

Kapitola 3

Metódy strojového učenia

V tejto kapitole uvádzame krátky úvod do problematiky strojového učenia (ML, *machine learning*) a popisujeme metódy, ktoré v práci používame. Uvedieme definíciu klasifikačnej úlohy, rozdiel medzi učením s učiteľom a učením bez učiteľa, všeobecný postup pri riešení úloh pomocou metód strojového učenia a postupy, ktoré nám pomáhajú vysporiadať sa s malým počtom údajov. Ďalej uvedieme ako sa pri strojovom učení reprezentujú reálne objekty. V neposlednom rade uvedieme miery, ktoré sa používajú pri evaluácii výsledkov a popíšeme metódy, ktoré budeme v tejto práci používať.

Úlohy strojového učenia sú často nasadzované na úlohy, ktoré už ľudia nie sú schopní riešiť manuálne. To môže byť spôsobené jednak tým, že objem údajov, ktoré musí úloha spracovávať, je obrovský a jednak tým, že úlohu je schopná riešiť manuálne len malá skupina ľudí (expertov). Riešenie takýchto úloh manuálne by potom bolo drahé (časovo aj finančne).

Pri spracovávaní prirodzených jazykov je veľa úloh možno riesiť práve pomocou strojového učenia. Pre rôzne úlohy sú vhodné rôzne metódy a algoritmy. Výber tej správnej metódy je kľúčovou časťou riešenia každej úlohy.

3.1 Reprezentácia objektov reálneho sveta v ML

V počítačovej lingvistike pod objektami reálneho sveta obvykle rozumieme vety, slová, fonémy a pod. Všetky tieto objekty musíme reprezentovať vo vhodnej forme zrozumiteľnej pre počítače. V strojovom učení reprezentujeme objekty pomocou vektorov rysov (*feature vectors*). Každá zložka vektora rysov predstavuje určitý rys, ktorým je objekt možné charakterizovať.

Rysy, ktoré tvoria zložky vektorov rysov, môžu byť číselné (čísla môžu byť prirodzené alebo reálne) alebo kategoriálne. Hlavným rozdielom medzi nimi je možnosť usporiadania číselných hodnôt a nemožnosť usporiadania kategoriálnych hodnôt.

Napríklad, každé podstatné meno môžeme reprezentovať vektorom rysov, v ktorom môžu jednotlivé zložky reprezentovať napríklad rod, číslo a pád podstatného mena. Rys rod bude zrejme kategoriálny, pretože môže nadobúdať len 3 hodnoty a tie medzi sebou nemajú definované žiadne usporiadanie. Číselným rysom, ktorý by mohol reprezentovať podstatné meno by mohla byť napríklad jeho frekvencia v korpuse.

Formálne môžme popísané objekty definovať nasledujúcim spôsobom:

- Nech reálne objekty sú prvkami množiny X: $x \in X$
- Nech množina A je množina rysov a $Values(a_i)$ je množina všetkých možných hodnôt rysu a_i. Nech $|A| = n$.
- Vektor rysov je potom usporiadaná n-tica $x = [x_1, x_2, x_3, ..., x_n]$, kde $x_i \in Values(a_i)$.

- Vektor rysov môžme potom chápať ako bod v n-rozmernom vektorovom priestore.

V našej práci budeme vety, ktorým bude musieť náš klasifikátor priradiť patterny reprezentovať tiež pomocou vektorov rysov. To, akými rysmi budeme vety reprezentovať, tvorí jadro tejto práce a je podrobne popísané v kapitole 6.

3.2 Klasifikačná úloha

Klasifikačná úloha je úloha, v ktorej vstupným objektom priraďujeme nejakú značku (*label*, *tag*) z predom definovanej množiny značiek (*tagset*). Príkladom takejto značky môže byť údaj o rode podstatného mena. Množina značiek by potom bola tvorená tromi prvkami (pre mužský, ženský a stredný rod). Takáto úloha môžeme byť riešená viacerými spôsobmi. Napríklad metódami strojového učenia, ľuďmi, alebo počítačmi, ktorým algoritmus riešenia popísali ľudia.

Formálne, je definovaná množina značiek Y. Nech funkcia $t : X \to Y$ je skutočná predikcia, ktorá označí objekt $x \in X$ správnou značkou. Riešením klasifikačnej úlohy je vytvorenie funkcie $c : X \to Y$, ktorá čo najlepšie aproximuje funkciu $t(x)$.

V prípade strojového učenia sa vedomosti o tom, ako klasifikovať jednotlivé objekty získavajú zo zadanej množiny objektov. Množina týchto objektov tvorí trénovacie údaje. Existujú 2 základné spôsoby, akými sa trénovacie údaje môžu použiť. Podľa toho, ktorý z týchto spôsobov metóda strojového učenia používa, ju môžeme zaradiť medzi metódy s učením s učiteľom (*supervised methods*) a metódy s učením bez učiteľa (*unsupervised methods*).

Pri strojovom učení s učiteľom sú trénovacie údaje tvorené objektami, ktoré už majú priradenú cieľovú značku. Cieľová znalosť sa teda získava z objektov, ktorým už bola priradená cieľová značka. Prepokladá sa, že táto značka bola priradená správne. Tento spôsob je najčastejší pri úlohách spracovania prirodzeného jazyka.

Formálne môžme definovať množinu údajov potrebných pre učenie s učiteľom takýmto spôsobom:

$$Data = \{[x, y]; x \in X, y \in Y\}$$

Trénovacie údaje môžeme potom definovať ako množinu D, pre ktorú platí, že $D \subset Data$. Testovacie údaje definujeme ako množinu D_{test}, pre ktorú platí $D_{test} \subset Data \wedge D \cap D_{test} = \emptyset$.

Strojové učenie bez učiteľa nevyžaduje, aby trénovacie údaje boli označkované cieľovými značkami. Výhodou tohto spôsobu je, že metóde môžeme poskytnúť viac trénovacích údajov (sú dostupné ľahšie, pretože nie je nutné, aby cieľovú značku priradzovali ľudskí experti). Na druhej strane metódy bez učiteľa majú vo všeobecnosti ťažšiu úlohu a sú zložitejšie, než metódy s učiteľom.

Formálne môžme definovať množinu údajov potrebných pre učenie bez učiteľa takýmto spôsobom:

$$Data = \{x; x \in X\}$$

Trénovacie a testovacie údaje môžeme potom definovať úplne rovnako ako v prípade učenia s učiteľom.

Pomer medzi veľkosťou trénovacích a testovacích údajov nie je pevne daný a pre rôzne úlohy sa môžu hodiť rôzne pomery. Vo všeobecnosti však platí, že čím viac inštancií použijeme pri trénovaní, tým vyššiu kvalitu bude mať získaný model. Čím viac údajov použijeme pri testovaní, tým viac bude získaný výsledok štatisticky signifikantnejší. Vo väčšine prípadov sa podiel testovacích údajov pohybuje medzi 10% a 30%.

Pri niektorých prípadoch sa z trénovacích údajov môže oddeliť ešte ďalšia množina údajov, ktoré nazývame **held-out** údaje. Používajú sa vo fáze ladenia parametrov modelu.

V mnohých prípadoch sa oba postupy pri riešení úloh kombinujú. Najskôr sa pomocou malého množstva označkovaných objektov a použitím metód s učiteľom vytvorí základná štruktúra vedomostí, ktorá sa následne pomocou metód bez učiteľa a veľkého množstva údajov ladí štatistickými metódami.

V našej práci budeme používať len metódy strojového učenia s učiteľom.

3.3 Metodológia riešenia úloh strojovým učením

Pri riešení úloh sa postupuje nasledujúcim spôsobom:

- Dostupné označkované údaje sa rozdelia na dve časti - trénovacie a testovacie údaje.
- Trénovacie údaje sa použijú pri trénovaní metód strojového učenia. Výsledkom je model, ktorý reprezentuje funkciu $c(x)$.
- Natrénovaný model sa použije na klasifikáciu testovacích údajov a úspešnosť modelu sa vyhodnotí pomocou niektorej z evaluačných mier.

Častým problémom pri riešení úloh strojového učenia s učiteľom je nedostatočné množstvo údajov. Ako sme už spomínali vyššie, získanie označkovaných inštancií je náročné časovo i finančne, pretože manuálne priradzovanie je schopná robiť len obmedzená skupina expertov. Jedným z možných riešení, ako zlepšiť kvalitu modelov je použiť **cross-validáciu**.

Pri cross-validácii sa všetky dostupné údaje náhodne rozdelia na n častí. Nasleduje n cyklov, v ktorých sa model testuje na n-tej časti údajov a trénuje na zvyšných častiach. Výsledné hodnotenie modelu je potom priemerom cez všetkých n cyklov. Výhodou tohto postupu je aj to, že údaje testujeme postupne na všetkých inštanciách, ktoré máme k dispozícii. Nevýhodou tohto postupu je jeho časová náročnosť, ktorá sa v porovnaní s klasickým postupom zvyšuje n-krát.

3.4 Evaluácia

Jednou z najdôležitejších častí celého procesu strojového učenia je správne vyhodnotenie výkonnosti natrénovaných modelov. Na vyjadrenie tejto výkonnosti sa používa niekoľko rôznych metrík. Najznámejšie a najpoužívanejšie sú definované pomocou matice konfúzie.

Matica konfúzie je definovaná nasledujúcou schémou:

		PP	
		OK	KO
PK	OK	*tp*	*fp*
	KO	*fn*	*tn*

Tabuľka 3.1: Matica konfúzie medzi pravdivou predikciou (PP) a predikciou klasifikátora (PK).

Matica a popísané miery boli pôvodne vytvorené pre úlohu vyhľadávania dokumentov. Riešením tejto úlohy je označiť dokument jednou z dvoch možných značiek ($Y = \{0, 1\}$) v závislosti od toho, či dokument bude vo výsledkoch vyhľadávania na zadaný dopyt, alebo nie.

Jednotlivé hodnoty v matici sú definované nesledujúcim spôsobom:

- *tp* (*true positive*) je počet (v skutočnosti) relevantných dokumentov, ktoré klasifikátor označil za relevantné
- *fp* (*false positive*) je počet (v skutočnosti) irelevantných dokumentov, ktoré klasifikátor nesprávne označil za relevantné
- *tn* (*true negative*) je počet (v skutočnosti) irelevantných dokumentov, ktoré klasifikátor označil za irelevantné
- *fn* (*false negative*) počet (v skutočnosti) relevantných dokumentov, ktoré klasifikátor nesprávne označil za irelevantné

Na základe týchto údajov sú definované miery Accuracy (A), Precision (P), Recall (R) a F-measure (F) nasledujúcim spôsobom:

$$A = \frac{tp + fn}{tp + fp + tn + fn}$$

$$P = \frac{tp}{tp + fp}$$

$$R = \frac{tp}{tp + fn}$$

$$F = \frac{1}{\alpha\frac{1}{P} + (1-\alpha)\frac{1}{R}}$$

Miera Accuracy (A) vyjadruje podiel správne klasifikovaných dokumentov voči všetkým dokumentom. Miera Precision (P) vyjadruje podiel skutočne relevantných dokumentov z tých, ktoré boli nájdené. Miera Recall (R) vyjadruje akú časť z relevantných dokumentov sa nám podarilo nájsť. Nakoniec miera F-measure kombinuje metriky P a R. Pomocou koeficientu α je možné špecifikovať, na ktorú mieru sa má klásť väčší dôraz.

3.5 Rozhodovacie stromy

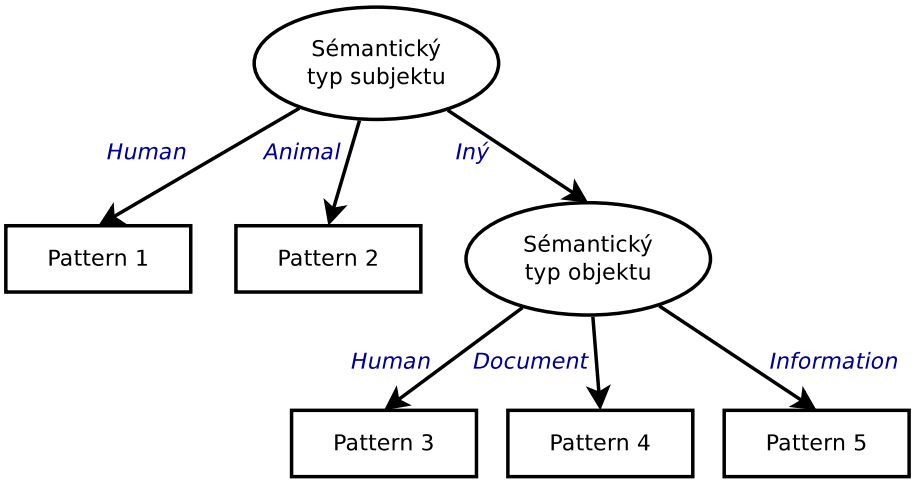
Rozhodovací strom je klasifikačný model, ktorý je založený na množine rozhodnutí, ktorá je uporiadaná do stromovej štruktúry.

Formálne môžeme rozhodovací strom definovať pomocou matematických štruktúr z teórie grafov:

- Graf $G(V, E)$ nazveme **strom**, ak G je súvislý a neobsahuje kružnicu.
- **Zakorenený strom** je orientovaný graf, ktorý je strom, a v ktorom definujeme jeden vrchol ako koreň a všetky hrany zorientujeme smerom od tohto koreňa k listom.
- **Rozhodovací strom** $G_{DT}(V_{DT}, E_{DT})$ je zakorenený strom, ktorého vnútorné uzly pomenujeme ako rozhodovacie a vonkajšie uzly ako listy.

Každý rozhodovací uzol reprezentuje jedno rozhodnutie algoritmu podľa vybraného rysu z množiny rysov $Attr$. Jednotlivé hrany, ktoré vedú z tohto uzla predstavujú možné hodnoty, ktoré môže rys nadobúdať. Listy rozhodovacieho stromu predstavujú prvky z množiny cieľových značiek Y, ktoré klasifikátor priradí inštancii, ktorej rysy vyhovujú všetkým podmienkam na ceste z koreňa do listu.

Rozhodovacie stromy môžu byť



Obr. 3.1: Grafické znázornenie rozhodovacieho stromu s dvomi rozhodovacími vrcholmi a s piatimi listami.

- Klasifikačné, kde cieľová množina značiek je kategoriálna
- Regresné, kde cieľová množina značiek je tvorená číslami (diskrétnymi alebo spojitými)

Rozhodovacie stromy majú niekoľko výhod, vďaka ktorým patria medzi najpopulárnejšie metódy strojového učenia. Sú ľahko pochopiteľné, zodpovedajú spôsobu, akým riešia úlohy ľudia. Predstavujú tzv. *white-box* – mechanizmus rozhodovania je ľahko vysvetliteľný a je presne dané, akým spôsobom rozhodovací strom dôjde k výslednej značke.

Sú tiež ľahko implementovateľné - nepotrebujú normalizované údaje ani množstvo pomocných premenných. V neposlednom rade, modely rozhodovacích stromov sú robustné a relatívne rýchle i na obrovských objemoch údajov.

Na obrázku 3.1 je graficky znázornený príklad rozhodovacieho stromu, ktorý by mohol byť použitý ako klasifikátor patternov. V prvom rozhodovacom uzle sa zisťuje, či sémantický typ subjektu testovanej inštancie je typu [[**Human**]] alebo [[**Animal**]]. Ak áno, klasifikátor okamžite priradí pattern 1 alebo pattern 2. Ak je subjekt iného sémantického typu, inštancia je nasmerovaná na druhý rozhodovací uzol, v ktorom sa zisťuje, či sémantický typ objektu je typu [[**Human**]], [[**Document**]] alebo [[**Information**]]. Na základe tohto rozhodnutia je inštancii priradený niektorý z patternov 3–5. Pre úplnosť uvádzame aj formálny zápis práve tohto rozhodovacieho stromu v podobe algoritmu 1.

Rozhodovací strom je výsledkom trénovacej fázy, počas ktorej je strom vytvorený na základe trénovacích údajov. Jednotlivé implementácie rozhodovacích stromov sa líšia v postupe, akým je rozhodovací strom indukovaný. Vo všeobecnosti však indukcia rozhodovacieho stromu spočíva v použití hladového výberu a princípu rozdeľ a panuj. Algoritmus tak pozostáva z dvoch častí:

1. V hladovej fáze algoritmus spočíta podmienené frekvencie distribúcie výstupných tried v závislosti na konkrétnych rysoch. Tieto distribúcie sú následne vyhodnotené pomocou určitých mier. Konkrétna miera, ktorá sa v algoritme používa, sa líši od konkrétnej implementácie. Rys, ktorý dosiahne najvyšiu hodnotu v danej miere, je vybraný ako rys, ktorý bude tvoriť rozhodovací uzol v strome.
2. Nasleduje fáza rozdeľ a panuj. Trénovacie údaje sú rozdelené do skupín podľa

Algoritmus 1 Formálny zápis algoritmu ilustrovaného na obrázku 3.1.

```
if Sémantický typ subjektu == Human then
  return Pattern 1
else if Sémantický typ subjektu == Animal then
  return Pattern 2
else
  if Sémantický typ objektu == Human then
    return Pattern 3
  else if Sémantický typ objektu == Document then
    return Pattern 4
  else if Sémantický typ objektu == Information then
    return Pattern 5
  end if
end if
```

možných hodnôt, ktoré môže nadobúdať rys vybraný v prvej fáze. Celá procedúra sa potom rekurzívne opakuje na každú z týchto skupín zvlášť. Každá vetva rozhodovacieho stromu teda pracuje s menším počtom trénovacích údajov než predošlé.

Rekurzia končí v prípade, ak nie je možné vybrať dostatočne dobrý rys, podľa ktorého by sa údaje mohli rozdeliť, alebo všetky údaje v skupine už majú rovnakú výslednú triedu.

Po trénovacej fáze zvyčajne prichádza na rad testovacia fáza. Postavený rozhodovací strom sa v nej použije na klasifikáciu testovacích inštancií. Rozhodovací proces pre každú inštanciu začína v koreni stromu. Koreň stromu je prvý rozhodovací uzol, v ktorom sa porovná hodnota definovaného rysu v danej inštancii a podľa nej sa inštancia presunie buď do listu, čím sa jej priradí výstupná trieda, alebo do ďalšieho rozhodovacieho uzlu, v ktorom sa testuje ďalší rys. Každá inštancia tak prechádza rozhodovacím stromom, kým neskončí v niektorom z listov.

3.5.1 Výber rysov do rozhodovacích uzlov

Ako sme už uviedli v predchádzajúcom texte, výber rysov, ktoré budú tvoriť rozhodovacie uzly je kľúčovou záležitosťou pre výkonnosť rozhodovacích stromov. Hlavným kritériom, ktorým sa výber rysov riadi, by malo byť pravidlo, že rozdelením údajov do vetiev stromu by malo dôjsť k tomu, že údaje vo vetve budú čistejšie v zmysle homogenity výstupných tried.

Formále môžeme pre každý rozhodovací uzol t rozhodovacieho stromu (keďže rozhodovací uzol zodpovedá práve jednému rysu z množiny rysov) a pre c výstupných tried definovať pravdepodobnostné rozdelenie $p(j|t)$, kde $j = 1, ..., c$, ktoré vypovedá o proporčnom rozdelení výstupných tried v danom uzle. Zrejme platí, že

$$\sum_{j=1}^{c} p(j|t) = 1$$

Pomocou tohto rozdelenia môžeme definovať mieru *nečistoty* $i(t)$, ako nezápornú funkciu c-premenných Φ:

$$i(t) = \Phi(p(1|t), p(2|t), ..., p(c|t)),$$

pre ktorú platia nasledujúce vlastnosti:

- Funkcia dosahuje maximum, ak sú výstupné triedy údajov v uzle rozdelené rovnomerne, tj. $\Phi(\frac{1}{c}, \frac{1}{c}, ..., \frac{1}{c}) = max$.

- Hodnota funckie je rovná nule, ak všetky údaje v uzle patria do jednej výstupnej triedy, tj. $\Phi(1,0,0,...,0) = \Phi(0,1,0,...,0) = ...\Phi(0,0,0,...,1) = 0$.

Pomocou funkcie $i(t)$ tak môžeme definovať vhodnosť rozdelenia s ako zmenu nečistoty údajov, ktorá nastala po rozdelení údajov v uzle t do uzlov t_L a t_R:

$$\Delta i(s,t) = i(t) - p_L \cdot i(t_L) - p_R \cdot i(t_R),$$

kde p_L a p_R vyjadrujú pomer, s akým sa údaje z uzlu t rozdelili do uzlov t_L a t_R.

Úlohou algoritmu, ktorý indukuje rozhodovací strom, je teda pre každý uzol nájsť také najvhodnejšie rozdelenie s z množiny možných rozdelení S, ktoré maximalizuje zmenu nečistoty $\Delta i(s,t)$ (a teda minimalizuje nečistotu):

$$\Delta i(s*,t) = \max_{s \in S} \Delta i(s,t)$$

Medzi štyri najznámejšie predpisy pre funkciu $i(t)$ patria nasledujúce funkcie. Ich názvy uvádzame v angličtine, pretože nie je známy žiaden slovenský ekvivalent:

Misclassification error

Funkcia $i(t)$ je definovaná predpisom

$$i(t) = 1 - \frac{1}{|t|} \sum_{[x,y] \in t} \delta_{y,\hat{k}} = 1 - p_t^{\hat{k}},$$

kde $\hat{k}$ predstavuje výstupnú triedu, ktorá je zastúpená v najvyššom počte, čo môžme formálne definovať pomocou predpisu

$$\hat{k} = \arg\max_{k \in Y} p_t^k = \arg\max_{k \in Y} (|t|^{-1} \sum_{[x,y] \in t} \delta_{y,k}).$$

V uvedenom predpise je Y množina výstupných tried a funkcia $\delta_{y,k}$ vracia 1, ak $y = k$, inak vracia 0. Misclassification error teda určuje nečistotu rozdelenia ako chybu, ktorej by sme sa dopustili, ak by sme všetky inštancie v danom rozhodovacom uzle ohodnotili najčastejšou výstupnou triedou.

Information gain

Funkcia $i(t)$ je definovaná predpisom

$$i(t) = -\sum_{j=1}^{c} \log(p(j|t)).$$

Nejedná sa o nič iné, ako o definíciu entropie a platí, že $i(t) = H(t)$. Information gain sa na distribúciu výstupných tried v danom rozhodovacom uzle pozerá ako na distribúciu náhodnej premennej, pre ktorú je možné spočítať entropiu. Tá splňuje všetky vlastnosti, ktoré požadujeme pre funkciu Φ:

- v prípade, že je distribúcia náhodnej premennej rovnomerná, entropia dosahuje najvyššiu hodnotu,
- v prípade, že distribúcia obsahuje len jednu dominantnú hodnotu, entropia sa rovná nule.

Môžeme tu vidieť paralelu k populárnej definícii entropie vo fyzike, ktorá vyjadruje mieru neusporiadania systému. Ako systém môžeme v našom prípade uvažovať distribúciu výstupných tried nad rozhodovacím uzlom.

Problémom tejto miery je fakt, že favorizuje rysy s väčším počtom možných hodnôt oproti rysom s malým počtom možných hodnôt. Jej výpočet totiž neberie do úvahy počet možných hodnôt jednotlivých rysov. Nasledujúca miera sa tento nedostatok snaží odstrániť.

Gain ratio

Na základe definície predpisu pre Information gain môžme definovať funkciu $\Delta i(s,t)$ nasledujúcim predpisom:

$$\Delta i(s,t) = i(t) - \sum_{s_j} p_j i(t_j) = H(t) - \sum_{s_j} \frac{|t_j|}{|t|} H(t_j).$$

Takto definovanú funkciu označujeme ako funkciu $Gain(s,t)$:

$$\Delta i(s,t) = Gain(s,t).$$

Pomocou funkcie $Gain(s,t)$ je definovaný ďalší spôsob vyjadrenia nečistoty uzlu:

$$i(t) = \frac{Gain(s,t)}{-\sum_{s_j} \frac{|t_j|}{|t|} \log(\frac{|t_j|}{|t|})},$$

kde t_j je množina inštancií, ktorých výstupná trieda je práve j.

Gini index

Funkcia $i(t)$ je definovaná predpisom

$$i(t) = \sum_{j \in Y, j \neq i} p(j|t)p(i|t) = 1 - \sum_{j \in Y} p^2(j|t).$$

Gini index by sme mohli slovne definovať ako očakávanú chybovosť. Pre uzol t najskôr $p(i|t)$ vyjadruje pravdepodobnosť, že náhodne vybraná inštancia z t bude mať výstupnú triedu i. Následne $p(j|t)$ vyjadruje pravdepodobnosť, že tá istá inštancia bude klasifikovaná triedou j. Výsledná hodnota je teda očakávaná miera zle klasifikovaných inštancií na danom rozhodovacom uzle t.

Všetky uvedené miery sú si do istej miery veľmi podobné. Existuje však medzi nimi niekoľko dôležitých rozdielov. Miery využívajúce entropiu (Information gain a Gain ratio) a Gini index

- sú viac diferencovateľné a umožňujú tak lepšiu numerickú optimalizáciu než Misclassification error,
- sú citlivejšie na zmenu distribúcie výstupných tried v rozhodovacom uzle, než Misclassification error.

3.5.2 Algoritmy vytvárania rozhodovacích stromov

Neformálne sme algoritmus vytvárania rozhodovacieho stromu popísali už v sekcii 3.5. Formálne ho popisujeme v algoritme 2, v ktorom je pseudokódom popísaná procedúra VytvorUzol. Procedúra vyžaduje 2 parametre:

- množinu inštancií trénovacích údajov D
- množinu rysov A, pomocou ktorých sú inštancie reprezentované

Procedúra pri svojom zavolaní vytvorí koreň stromu. Vyberie na to najvhodnejší rys a_i (napríklad na základe niektorej z vyššie uvedených mier) a rekurzívne volá samu seba, aby získala všetky podradené uzly. Každému volaniu procedúry pritom ako parametre odovzdá podmnožinu trénovacích údajov D_i, ktorá sa zhoduje v hodnote zvoleného rysu a_i a množinu rysov A_i, v ktorej sú všetky rysy, ktoré ešte neboli použité v rozhodovacích uzloch.

Algoritmus 2 Procedúra VytvorUzol(D, A)

```
Vytvor uzol t
if všetky inštancie majú rovnakú výstupnú triedu j then
  return uzol t ako list, výstupná trieda pre uzol t bude j
else if množina rysov A je prázdna then
  return uzol t ako list, výstupná trieda pre uzol t bude (napríklad) najčastejšia
  trieda v množine inštancií D
else if množina inštancií D je prázdna then
  return uzol t ako list, výstupná trieda pre uzol t bude (napríklad) najčastejšia
  trieda v množine inštancií D
else
  Vyber rys a_i, ktorý minimalizuje nečistotu maximálne
  Vytvor množinu rysov A_i, pre ktorú platí A_i = A \ {a_i}
  for h ∈ Values(a_i) do
    vytvor množinu inštancií D_i, v ktorej budú všetky inštancie z množiny D, v kto-
    rých rys a_i nadobúda hodnotu h
    získaj uzol v volaním VytvorUzol(D_i, A_i)
    uzol v použi ako uzol, ktorý je kladným výsledkom testu, či rys a_i má hodnotu
    h
  end for
end if
```

Uvedená procedúra predstavuje základný princíp algoritmu, ktorý sa v rôznych implementáciách môže líšiť o ďalšie vylepšenia.

V nasledujúcej časti práce uvádzame niekoľko najznámejších algoritmov na indukciu rozhodovacích stromov.

ID3 Algoritmus

Algoritmus je asi najznámenším algoritmom pre tvorbu rozhodovacích stromov. Bol publikovaný v [22] a umožňuje vytvoriť rozhodovací strom, ktorý pracuje s dvojprvkovou množinou výstupných tried.

Princíp algoritmu je totožný so všeobecným algoritmom popísaným vyššie. Ako miera určujúca rys, ktorý sa vyberie do rozhodovacieho uzlu, sa používa Information gain.

C4.5 Algoritmus

Algoritmus bol popísaný v [23]. Je založený na algoritme ID3 a obsahuje niekoľko zlepšení:

- algoritmus dokáže pracovať s hodnotami rysov, ktoré sa nevyskytovali v trénovacích inštanciách,

- je možné použiť aj rysy so spojitými číselnými hodnotami,
- namiesto miery Information gain používa mieru Gain ratio,
- v priebehu algoritmu sa z trénovacích inštancií oddelí interne malá časť inštancií, ktoré sa používajú na testovanie v priebehu algoritmu. Po každom pridaní vrcholu je vyhodnocované, či sa úspešnosť rozhodovacieho stromu zvýšila.

C5.0 Algoritmus

Najnovšia verzia algoritmu C4.5 bola publikovaná v [24]. Vylepšuje výkonnosť algoritmu C4.5 najmä:

- vytváraním niekoľkých klasifikátorov súčasne pomocou boostingu[1]; a
- systémom penalizácie chybne klasifikovaných inštancií.

Keďže sa jedná o komerčný produkt, algoritmus nie je známy do podrobných detailov.

3.6 Najbližší susedia

Algoritmus strojového učenia s názvom k najbližších susedov [25] patrí do skupiny algoritmov, ktoré sa označujú ako algoritmy založené na príkladoch (*instance-based*).

Dokážu modelovať diskrétne aj spojité výstupné množiny. Niekedy sa označujú aj ako *lenivé* metódy (*lazy methods*). Na rozdiel, napríklad od rozhodovacích storomov, ktoré trénovacie údaje používajú na zostavenie rozhodovacieho stromu, lenivé metódy neindukujú žiadne modely, ale jednoducho si všetky trénovacie inštancie uchovávajú a samotný algoritmus je spustený až v testovacej fáze, v ktorej je ich úlohou klasifikovať testovaciu inštanciu. Algoritmus v tejto fáze použije vhodné trénovacie inštancie, ktoré sú testovacej inštancii najpodobnejšie. Klasifikácia je teda určená na základe lokálnej podobnosti.

Okrem algoritmu k najbližších susedov patria do skupiny algoritmov založených na príkladoch aj

- k najbližších susedov s váženou vzdialenosťou; a
- lokálne vážená regresia.

Základom všetkých algoritmov založených na podobnosti je presná matematická definícia podobnosti. Keďže inštancie sa v strojovom učení reprezentujú pomocou vektorov rysov, podobnosti sú definované ako predpisy, ktoré pre každé 2 vektory rysov určia číselné vyjadrenie podobnosti v podobe vzdialenosti medzi vektormi. Čím je vzdialenosť menšia, tým sú si vektory podobnejšie.

Vzdialenosť medzi vektormi rysov sa zvykne označovať tiež ako metrika. Formálne, pre n-prvkovú množinu rysov A a vektory rysov x a y definujeme predpis, ktorý vráti nezáporné reálne číslo vyjadrúce vzdialenosť vektorov x a y.

Pre vektory rysov, v ktorých rysy nadobúdajú číselné hodnoty sa najčastejšie používa Euclidovská metrika, ktorá je všeobecne známa z geometrie. Jej predpis je

$$E(x, y) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}$$

[1] Boosting podrobne popisuje v časti 3.8 tejto práce.

Okrem Euclidovskej metriky je známa aj Manhattanská metrika [26], ktorá je menej výpočtovo náročná:

$$M(x,y) = \sum_{i=1}^{n} |x_i - y_i|$$

Obe tieto metriky sú ale definované len pre rysy, ktoré nadobúdajú číselné hodnoty. Pre rysy kategoriálne bola definovaná metrika Value Difference Metric (VDM), ktorá jednoducho, pomocou funkcie $\delta_i(x_j, y_j)$ definovanej pre každý rys i tak, že vracia hodnotu 1, ak $x_j = y_j$, inak vracia 0. Metrika VDM je potom definovaná predpisom

$$VDM(x,y) = \sum_{i=1}^{n} \delta_i(x_i, y_i).$$

Pomocou definovaných metrík je teda možné presne matematicky vyjadriť vzdialenosť medzi jednotlivými vektormi rysov. Algoritmus k najbližších susedov tieto metriky využíva na to, aby k zadanej testovacej inštancii vybral z množiny trénovacích inštancií k vektorov, ktoré sú k testovacej inštancii najbližšie. Na základe týchto inštancí potom algoritmus priradí výslednú značku, resp. hodnotu pre testovaciu inštanciu.

Pre klasifikačnú úlohu sa najčastejšie vyberie najčastejšia výstupná trieda, ktorá sa medzi k najbližšími susedmi objavila. Formálne to môžeme zapísať ako funkciu $\hat{f}(x)$, pre ktorú platí

$$\hat{f}(x) = \arg\max_{y \in Y} \sum_{i=1}^{k} \delta(y, c(l_i)),$$

kde $l_1, l_2, ..., l_k$ je k najbližších susedov testovacej inštancie x, funkcia $c(l_i)$ vráti výstupnú triedu, ktorú má priradenú inštancia l_i a funkcia $\delta(y, c(l_i))$ vráti 1, ak $y = c(l_i)$, inak vráti 0.

Pre regresnú úlohu sa môže výstupná hodnota spočítať napríklad ako aritmetický priemer výstupných hodnôt k najbližších susedov:

$$\hat{f}(x) = \frac{1}{k} \sum_{i=1}^{k} c(l_i)$$

V základnej verzii algoritmu sa na výslednej značke podieľa každá z k trénovacích inštancií rovnakou váhou. To v praxi nemusí byť ideálne, pretože z nájdených k susedov môžu byť niektoré inštancie vzhľadom k testovacej inštancii oveľa bližšie, než iné.

Vylepšená verzia algoritmu preto prichádza s vážením dôležitosti jednotlivých k trénovacích inštancií a to podľa vzdialenosti od testovacej inštancie. Trénovacia inštancia má tým väčšiu váhu, čím bližšie je k testovacej inštancii. Formálne môžme definovať váhovú funkciu $w(x, l_i)$, ktorá pre testovaciu inštanciu x a trénovaciu inštanciu l_i vráti váhu, s akou má byť trénovacia inštancia l_i použitá pri klasifikácii x. Váhová funkcia môže byť definovaná napríklad predpisom

$$w(x, l_i) = \frac{1}{d(x, l_i)^2},$$

kde d je metrika použitá v algoritme.

Funkcia $\hat{f}(x)$ pre klasifikačnú úlohu môže mať potom tvar

$$\hat{f}(x) = \arg\max_{y \in Y} \sum_{i=1}^{k} w(x, l_i)\delta(y, c(l_i)).$$

V prípade regresnej úlohy by mohla byť funkcia $\hat{f}(x)$ definovaná napríklad vzťahom

$$\hat{f}(x) = \frac{\sum_{i=1}^{k} w(x, l_i) c(l_i)}{\sum_{i=1}^{k} w(x, l_i)}$$

V algoritmoch, ktoré vplyv trénovacích inštancií vážia, potom nie je žiaden problém zvoliť za k počet všetkých trénovacích inštancií. Najvzdialenejšie inštancie majú potom malý, ale predsa len nenulový vplyv na výsledok klasifikácie.

Algoritmy s využitím najbližších susedov majú nevýhodu v porovnaní s inými algoritmami spočívajúcu v ich odlišnej výpočtovej zložitosti. Kým rozhodovacie stromy potrebujú výpočtový výkon pri trénovacej fáze, lenivé metódy potrebujú čas najmä v testovacej fáze. Potrebujú tiež pracovať naraz so všetkými trénovacími údajmi, čo pri veľkých počtoch údajov môže vyústiť v značne pomalú implementáciu. Tento problém je možné efektívne vyriešiť pomocou nástrojov na indexovanie trénovacích inštancií (napríklad pomocou tzv. *k-D stomov*).

Ďalší problém, ktorý musia algortimy najbližších susedov riešiť, je *prekliatie dimenzionality*. Vo vektoroch rysov, ktoré sú tvorené príliš veľkým počtom rysov, sa môže stať, že niektoré rysy neprinášajú žiadnu informáciu o tom, ako má byť klasifikovaná výstupná hodnota. Naopak, môžu vďaka svojim veľkým hodnotám skresliť metriku tak, že klasifikátor bude mať veľmi nízku výkonnosť. Problém s veľkým počtom dimenzií nie je len problémom lenivých metód, musí sa riešiť napríklad aj v algoritmoch SVM, ktoré popisujeme ďalej v tejto práci. Medzi najčastejšie spôsoby, pomocou ktorých je možné zlepšiť výkonnosť klasifikátora, patrí vhodne obmedziť množinu rysov A a hodnoty jednotlivých rysov *normalizovať*. Pri normalizácii sa všetky číselné rysy zobrazia do intervalu $[0, 1]$, pričom sa zachová pomer medzi jednotlivými hodnotami. Zaistí sa tým, že všetky rysy budú mať v predpise pre metriku rovnakú váhu.

3.7 Podporné vektory

Princípom algoritmu podporných vektorov (*Support Vector Machine* – SVM) je vytvoriť klasifikátor pre dvojprvkovú výstupnú množinu ($Y = \{-1, 1\}$). Myšlienkou je nájsť predpis nadroviny, ktorá vo vektorovom priestore rozdelí inštancie do dvoch nadrovín, pričom každej nadrovine zodpovedá jedna výstupná trieda.

Autorom algoritmu je Vladimir Vapnik, ktorý ho puplikoval v 60-tych rokoch 20. storočia. Rozmach algoritmu prišiel ale až v 90-tych rokoch [27, 28]. Dovtedy totiž neboli známe kernelové funkcie, ktoré by umožňovali vytvoriť aj klasifikátory, ktoré by rozdelili inštancie ich nelineárnym zobrazením do iného vektorového priestoru.

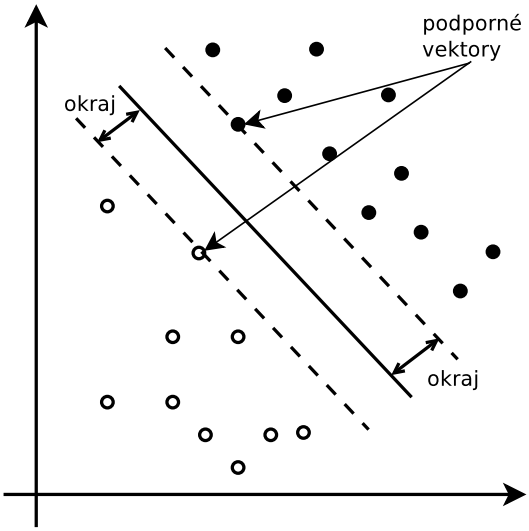
V nasledujúcom texte popíšeme princíp lineárneho klasifikátoru a načrtneme základné myšlienky nelineárnych klasifikátorov.

3.7.1 Lineárny klasifikátor

Nech $\mathcal{X}$ je n-rozmerný vektorový priestor inštancií a $c : \mathcal{X} \to \{-1, 1\}$ je funkcia, ktorá ohodnotí výstupnou triedou každú trénovaciu inštanciu. Vo vektorovom priestore musí byť definovaný skalárny súčin.

Cieľom lineárneho klasifikátora je vytvorenie nadroviny (jej rozmer bude $n - 1$), ktorá rozdelí inštancie vo vektorovom priestore do dvoch oblastí. V každej z oblastí sa budú nachádzať inštancie s rovnakou hodnotou funkcie c.

Pojmom okraj (*margin*) budeme označovať vzdialenosť nadroviny od najbližieho bodu (trénovacej inštancie) a to pre oba vzniknuté polpriestory. Cieľom klasifikátora je zo všetkých možných nadrovín, ktoré rozdeľujú inštancie podľa výstupnej triedy, a ktorých je nekonečné množstvo, vybrať takú, ktorá maximalizuje oba okraje.



Obr. 3.2: Lineárny klasifikátor s optimálnou nadrovinou

Voľba takejto nadroviny sa zdá byť intuitívne správna – jedná sa o *najbezpečnejšiu* nadrovinu. Pri chybne zvolenej nadrovine existuje najväčšia šanca, že inštancia bude správne klasifikovaná a naopak, pri správne zvolenej nadrovine je zabezpečená najväčšia robustnosť na šum v trénovacích inštanciách. Teoreticky je táto úvaha podporená Vapnik-Chervonenského teóriou [29] i empirickými výsledkami.

Uvažujme teraz prípad, kde vektorový priestor $\mathcal{X} = \mathbb{R}^2$. Na obrázku 3.2 uvádzame príklad nadroviny (pre dvojrozmerný vektorový priestor je nadrovina tvorená priamkou) s maximálnym okrajom. Na obrázku 3.3 zasa príklad, v ktorom rozdelenie priestoru nadrovinou nie je optimálne.

Formálne, každú nadrovinu môžeme definovať vektorom $\boldsymbol{w}$ a číslom b pomocou predpisu

$$\boldsymbol{w} \cdot \boldsymbol{x} - b = 0,$$

kde symbol $\cdot$ značí skalárny súčin.

Okraj takto definovanej nadroviny potom môžme definovať pomocou bodov $\boldsymbol{x}$, ktoré vyhovujú rovnici

$$\boldsymbol{w} \cdot \boldsymbol{x} - b \in \{-1, 1\}.$$

Hľadáme nadrovinu, ktorá správne klasifikuje všetky trénovacie údaje, teda bude platiť pre všetky inštancie $\boldsymbol{x_i}$, také, že $c(\boldsymbol{x_i}) = 1$

$$\boldsymbol{w} \cdot \boldsymbol{x_i} - b \geq 1$$

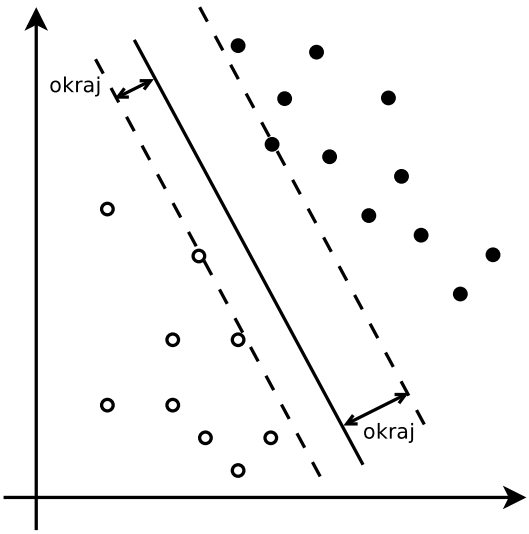
a pre všetky inštancie $\boldsymbol{x_i}$, také, že $c(\boldsymbol{x_i}) = -1$ bude platiť

$$\boldsymbol{w} \cdot \boldsymbol{x_i} - b \leq -1$$

Daná nadrovina zároveň musí mať maximálny okraj. Na obrázku 3.2 môžeme vidieť, že nadrovina závisí na bodoch, ktoré sú k nej najbližšie a ktorými prechádza hranica okraja. Tieto inštancie sa nazývajú podporné vektory. Pre všetky podporné vektory platí predpis

$$\text{abs}(\boldsymbol{w} \cdot \boldsymbol{x_i} - b) = 1.$$

Nech body $\boldsymbol{x_1}$ a $\boldsymbol{x_2}$ sú dva podporné vektory, a ležia v rôznych polpriestoroch vymedzených nadrovinou. Bez ujmy na všeobecnosti, nech $c(\boldsymbol{x_1}) = 1$ a $c(\boldsymbol{x_2}) = -1$.



Obr. 3.3: Lineárny klasifikátor s neoptimálnou nadrovinou

Keďže sa jedná o podporné vektory, platia pre ne rovnice

$$\boldsymbol{w} \cdot \boldsymbol{x_1} - b = 1,$$

$$\boldsymbol{w} \cdot \boldsymbol{x_2} - b = -1.$$

Rovnice môžeme skombinovať a získame:

$$\boldsymbol{w} \cdot (\boldsymbol{x_1} - \boldsymbol{x_2}) = 2$$

$$\frac{\boldsymbol{w}}{\|\boldsymbol{w}\|} \cdot (\boldsymbol{x_1} - \boldsymbol{x_2}) = \frac{2}{\|\boldsymbol{w}\|}.$$

Z predpisu vyplýva, že ak chceme maximalizovať okraj, musíme minimalizovať $\|\boldsymbol{w}\|$ so zachovaním podmienky

$$c(\boldsymbol{x_i}) \cdot (\boldsymbol{w} \cdot \boldsymbol{x_i} - b) \geq 1.$$

Takto definovanú úlohu je možné riešiť ako optimalizačnú úlohu pomocou Lagrangeových multiplikátorov.

3.7.2 Nelineárny klasifikátor

Existujú prípady, v ktorých inštancie vo vektorovom priestore nie je možné oddeliť lineárnou nadrovinou. Priamočiare riešenie je vytvoriť predpis pre zložitejší geometrický útvar. Algoritmus podporných vektorov ale využíva iný prístup – vhodnou nelineárnou transformáciou inštancií do rôznych vektorových priestorov sa snaží objaviť taký vektorový priestor, v ktorom existuje nadrovina, ktorá dokáže správne separovať všetky inštancie.

Podobne, ako v prípade lineárneho klasifikátoru, i v prípade nelineárneho klasifikátora je na vektorové priestory kladená podmienka existencie definície skalárneho súčinu a to pre východzí i cieľový vektorový priestor. Definícia skalárneho súčinu v oboch vektorových priestoroch umožňuje jednoduchým spôsobom definovať tzv. kernelovú funkciu, ktorá umožňuje v jednom predpise vyjadriť transformáciu vektorov τ a skalárny súčin:

$$k(\boldsymbol{x_1}, \boldsymbol{x_2}) = \tau(\boldsymbol{x_1}) \cdot \tau(\boldsymbol{x_2}).$$

Kernelová funkcia nemusí dimenziu vektorového priestoru len znižovať. Príkladom je transformácia, ktorá vektor z priestoru $\mathbb{R}^2$ zobrazuje do priestoru $\mathbb{R}^3$:

$$\tau(x_1, x_2) = (x_1, x_2, \sqrt{x_1^2 + x_2^2})$$

Na záver uvádzame niekoľko príkladov najčastejších kernelových funkcií:

- **Polynomiálne**: $k(x_1, x_2) = (x_1 \cdot x_2) + \theta$
- **Sigmoid**: $k(x_1, x_2) = \tanh(\kappa(x_1 \cdot x_2) + \theta)$
- **Gaussova radiálna báza**: $k(x_1, x_2) = \exp(-\frac{\|x_1 - x_2\|}{2\sigma^2})$

3.8 Adaboost

Adaboost je skrátené pomenovanie pre tzv. Adaptívny boosting. Algoritmus formulovali Yoav Freund a Robert Schapire v [30]. Nejedná sa o klasický algoritmus strojového učenia, ale o tzv. meta-algoritmus. Môže byť použitý v spojení s klasickými algoritmami strojového učenia a výrazne zlepšiť ich výkonnosť. Hlavnou myšlienkou je vygenerovať sekvenciu modelov získaných klasickými algoritmami. Jednotlivé modely v sekvencii sú ladené tak, aby eliminovali chyby, ktorých sa dopustili predchádzajúce modely.

Medzi výhody Adaboostu patrí predovšetkým jeho odolnosť na šum v trénovacích údajoch a na inštancie, ktoré sú v trénovacích údajoch veľmi málo podobné ostatným (tzv. outliers). Existujú však prípady, v ktorých môže byť Adaboost náchylný na pretrénovanie viac ako klasické algoritmy.

Adaboost generuje a volá nový slabý klasifikátor v každom cykle $t = 1, ..., T$. Po každom použití klasifikátora v iterácii t na inštanciu x_i sa prepočíta váha $D_t(i)$. Tá indikuje dôležitosť inštancie v množine trénovacích údajov. V každom z cyklov je dôležitosť nesprávne klasifikovaných inštancií zvyšovaná a dôležitosť správne klasifikovaných inštancií znižovaná. Tým je zabezpečené, že nový klasifikátor bude viac zameraný na inštancie, ktoré doteraz nedokázal správne klasifikovať.

Formálne si algoritmus popíšeme na príklade binárneho klasifikátora. Nech

$$(x_1, y_1), ..., (x_m, y_m),$$

kde $x_i \in X$ a $y_i \in Y$, kde $Y = \{-1, 1\}$, sú inštancie v množine trénovacích údajov. Nech počet iterácií algoritmu je T. V prvej iterácii sú všetky inštancie rovnocenné a váha $D_1(i)$ sa rovnomerne rozdelí medzi všetky inštancie:

$$D_1(i) = \frac{1}{m}, i = 1, ..., m$$

V každej iterácii $t = 1, ..., T$ sa potom postupuje tak, že z množiny možných klasifikátorov $\mathcal{H}$ sa vyberie taký klasifikátor h_t, ktorý maximalizuje rozdiel váženej chyby modelu ϵ_t a hodnoty $0,5$. Pre výpočet váženej chyby ϵ_t sa používa váhová funkcia D_t:

$$h_t = \arg\max_{h_t \in \mathcal{H}} |0,5 - \epsilon_t|$$

$$\epsilon_t = \sum_{i=1}^{m} D_t(i)\delta(y_i, h_t(x_i)),$$

kde $\delta(x, y)$ je funkcia, ktorá vráti 1, ak $x = y$ a 0 inak.

Následne sa pomocou váhovej funkcie D_t a modelu h_t spočíta nová váhová funkcia D_{t+1}, ktorá sa použije v ďalšej iterácii. Nech $\alpha_t \in \mathbb{R}$, typicky

$$\alpha_t = \frac{1}{2} \ln \frac{1 - \epsilon_t}{\epsilon_t}.$$

Potom

$$D_{t+1}(i) = \frac{D_t(i) \exp(-\alpha_t y_i h_t(x_i))}{\sum_{j=1}^{m} D_t(j) \exp(-\alpha_t y_j h_t(x_j))}$$

Takto definovaná váhová funkcia je zároveň pravdepodobnostné rozdelenie, tj. súčet všetkých hodnôt D_t sa rovná 1.

Výsledný klasifikátor H priradí testovacej inštancii značku podľa predpisu

$$H(x) = \text{sign}(\sum_{t=1}^{T} \alpha_t h_t(x)).$$

V experimentoch popisovaných v tejto práci sme sa rozhodli používať algoritmus Adaboost, ktorý obsahuje sekvenciu niekoľkých rozhodovacích stromov.

Kapitola 4

Automatické rozpoznávanie sémantiky slovies

V tejto kapitole uvádzame prehľad súvisiacich prác v oblasti automatického rozpoznávania sémantiky slovies. V prvej časti prezentujeme prehľad tradičných prístupov k WSD. V druhej časti uvádzame rešerš dizeračnej práce [37], ktorá sa zaoberá automatickou klasifikáciou valenčných rámcov. V poslednej časti uvádzame rešerš diplomovej práce [14], ktorá sa zaoberala úlohou vytvorenia pravidlového klasifikátora patternov PDEV.

4.1 Tradičná Word Sense Disambiguation (WSD)

Ako sme už uviedli v úvodne práce, WSD je klasickým otvoreným problémom od počiatkov počítačovej lingvistiky. Problémom sa vedci zaoberajú od 40-tych rokov, ktoré predstavujú ranné doby strojového prekladu.

Výskum problému WSD má bohatú históriu. V 70-tych rokoch bola WSD súčasťou interpretačných systémov v obore umelej inteligencie. Tento prístup však čoskoro narazil na nutnosť ručného definovania pravidiel a chýbajúcu automatickú akvizíciu vedomostí. V 80-tych rokoch umožnilo ďalší rozvoj WSD publikovanie veľkých lexikálnych zdrojov v elektronickej podobe. V 90-tych rokoch sa na WSD prvý krát použili metódy strojového učenia. Problém WSD sa tak stáva paradigmom pre aplikáciu metód strojového učenia s učiteľom.

Podľa [1, 36] v minulom desaťročí dosiahli používané techniky plató a pozornosť vedcov sa v súčasnosti presúva na riešenie otázky významovej granularity, adaptácie na rôzne domény, či kombinovanie rôznych existujúcich metód. Najlepšie výsledky sú dosahované metódami strojového učenia s učiteľom.

Tradičné prístupy k WSD možno podľa [1] rozdeliť na dve skupiny. Cieľom prvej je dokonalé pochopenie vedomostí o okolitom svete, ktoré budú použité na determinovanie zmyslu slov. Tento prístup nebol v minulosti príliš úspešný, predovšetkým pre chýbajúce zdroje vedomostí v počítačom spracovateľnej podobe [31]. Druhá skupina prístupov sa nesnaží o pochopenie textu. Namiesto toho vyvíjajú techniky, pomocou ktorých je možné z rozsiahlych trénovacích údajov extrahovať pravidlá, pomocou ktorých rozlíšia požadované významy. Aj ked sa môže zdať, že táto metóda nie je tak silná ako metódy prvej skupiny, v skutočnosti sú najlepšie výsledky dosahované práve v tejto skupine.

Existujú štyri tradičné prístupy k riešeniu WSD.

Slovníkové metódy a znalostné modely. Sú založené primárne na slovníkoch, tezauroch a znalostných lexikónoch, väčšinou bez použitia korpusov. Ako najprínosnejší algoritmus uvádza [1] Leskov algoritmus [32]. Je založený na predpokladoch, že slová použité v definíciách významov v slovníkoch sa často vyskytujú aj v kontexte cieľového

slova. Slovu algoritmus priradí taký význam, ktorého slovný popis sa najviac prekrýva s kontextom slova.

Ďalšie prístupy v tejto skupine pracujú napríklad s grafovými algoritmami, ktoré boli vyvinuté ešte v časoch riešenia WSD pomocou umelej inteligencie. Využívajú napríklad WordNet [11] a vytvárajú grafové štruktúry, v ktorých sú spojené slová na základe určitých parametrov, z ktorých sa počíta ich sémantická podobnosť. [33]

Strojové učenie s učiteľom. Prístupy predpokladajú existenciu sémanticky anotovaných korpusov, ktoré slúžia ako trénovacie údaje. V súčasnosti možno povedať, že všetky známe algoritmy strojového učenia už boli na riešenie WSD použité. Medzi najužitočnejšie algoritmy podľa [1] patria Podporné vektory (SVM) a algoritmy založené na príkladoch (*instance-based*). Oba algoritmy používame v experimentoch aj v tejto diplomovej práci.

Ako nevýhodu tohto prístupu môžeme uviesť práve nutnosť existencie anotovaných korpusov, ktorá je chápaná ako úzke hrdlo v celom procese.

Strojové učenie s polovičným učiteľom. Nevýhodu v podobe nutnosti existencie sémanticky anotovaných korpusov sa snaží prekonať prístup, v ktorom sa využívajú jednak anotované a jednak neanotované údaje. Typickým príkladom algoritmu je Yaroskeho algoritmus [34]. Hlavnou ideou je predpoklad, že v jednom dizkurze, resp. v jednej kolokácii sa význam cieľového slova nemení.

Ďalšou metódou v tejto skupine je metóda bootstrappingu. Klasifikátor natrénovaný pomocou malej množiny ručne anotovaných údajov anotuje veľký korpus. Najdôverihodnejšie klasifikácie sú v ďalšom kole použité na opätovné natrénovanie klasifikátora. Proces pokračuje, kým nie je automaticky anotovaný celý korpus, resp. kým nie je dosiahnutý maximálny počet opakovaní.

Strojové učenie bez učiteľa. Podľa [1] je tento prístup najväčšou význou pre WSD výskumníkov. Opäť sa tu využíva predpoklad, že slová majú v podobných kontextoch rovnaké významy. Na základne podobnosti kontextu sa aplikujú algoritmy zhlukovej analýzy. [35] Nevýhodou tohto prístupu je nutnosť mapovania vytvorených zhlukov do definícií významov popísaných v slovníkoch.

4.2 Automatická klasifikácia valenčných rámcov

Dizertačná práca [37] sa zaoberá podobnou úlohou, akú riešime v našej práci – vytvára automatický klasifikátor, ktorý každej vete priradí jeden z definovaných valenčných rámcov. Klasifikátor funguje pre češtinu a namiesto patternov používaných v PDEV pracuje s valenčnými rámcami.

Koncepcia valenčných rámcov, ktorá bola definovaná v [38], [39] a [40], vyjadruje schopnosť lexikálnej jednotky viazať iné lexikálne jednotky do syntaktických štruktúr. Valencia, ktorej názov je odvodený od podobného javu v chémii, bola popísaná pre niekoľko slovných druhov – pre podstatné a prídavné mená, príslovky a predovšetkým pre slovesá. Práve pre slovesá je valencia popísaná najpodrobnejšie a na Ústave formálnej a aplikovanej lingvistiky boli vytvorené 2 valenčné slovníky – VALLEX a PDT-VALLEX.

Valencia sa v teórii valenčných rámcov zachytáva do rámcov, ktoré definujú schopnosť slovesa viazať na seba iné lexikálne jednotky. Autor v jednej z kľúčových kapitol práce ukazuje, akým spôsobom je možné reprezentovať významy slovesa pomocou rôznych valenčných rámcov. Z tohto pohľadu je jasná podobnosť medzi klasifikátorom, ktorý sa pokúšame vytvoriť v tejto práci a klasifikátorom, ktorý vytvoril autor.

Oba slovníky (VALLEX/PDT-VALLEX a PDEV) sa snažia zachytiť významy slovies pomocou popísania kontextu slovesa. Kým valenčné rámce sú popísané predovšetkým syntakticky, hlavnou časťou definícií patternov v PDEV sú argumenty popísané pomocou sémantických typov a rolí. PDEV je teda viac zameraný na sémantickú zlož-

ku. Neváhame preto povedať, že má lepšie predpoklady na zachytenie významu ako valenčné rámce.

V ďalších častiach práce autor detailne popisuje metódy strojového učenia, ktoré pri svojich experimentoch používal a predovšetkým, z akých jazykových zdrojov čerpal údaje, ktorými reprezentoval jednotlivé inštancie vo vektoroch rysov. Ako základný zdroj, z ktorého autor čerpal definície valenčných rámcov poslúžili slovníky VALLEX a PDT-VALLEX. Zdrojom označkovaných inštancií sa stal korpus VALEVAL. Korpus obsahuje pre každé zo 109 slovies presne 100 ručne anotovaných viet.

Autor použil na reprezentáciu inštancií niekoľko druhov rysov:

- **Morfologické rysy**. Autor použil pozičný tagset používaný v PDT a pre každú inštanciu použil morfologické tagy slov z okolia slovesa. Veľkosť okolia slovesa vyjadruje oknom s veľkosťou n. Vo väčšine experimentov je $n = 5$, čo znamená, že sa používajú morfologické tagy cieľového slovesa a dvoch slov naľavo a napravo od slovesa. V práci autor popisuje aj experimenty s ďalšími veľkosťami okna.

- **Syntaktické rysy**. Autor použil automatický parser pre češtinu, z výstupu ktorého extrahoval niekoľko rôznyh rysov. Jedná sa napríklad o prítomnosť reflexívnych zámen si a se, prítomnosť podriadených, alebo naopak, nadriadených slovies vzhľadom k cieľovému slovesu, pády podstatných mien, zámen a prídavných mien a pod.

- **Idiomatické rysy**. Niektoré frazeologizmy determinujú význam slovies, preto sa autor rozhodol, že sa pokúsi tieto javy a významy slovies tiež zachytiť a rozlišovať. Idiomatické konštrukcie sú popísané v slovníku VALLEX. Všetky tieto konštrukcie autor prebral ako booleanovské rysy – prítomnosť konštrukcie vo vete je reprezentovaná hodnotou *true*, neprítomnosť zasa naopak hodnotou *false*.

- **Rysy vyjadrujúce živostnosť**. Životnosť je jednou z gramatických charakteristík podstatných mien a zámen. Autor použil rys životnosti na vyjadrenie, či argumentom slovesa je živá bytosť alebo nie. Hodnoty týchto rysov čerpal z morfologickej analýzy pre češtinu, ktorá pri lemmatizácii dokáže rozpoznať vlastné mená osôb a tiež z detailných slovných poddruhov definovaných pre zámená v češtine.

- **Sémantické rysy**. Autor použil českú verziu WordNetu na zaradenie podstatných mien nachádzajúcich sa vo vete do sémantickej ontológie definovanej pre projekt EuroWordNet. Táto ontológia definuje 64 synsetov, ktoré tvoria vrchol hyperonymickej hierarchie synsetov WordNetu. Na základe relácie hypernonymie sa pokúsil zaradiť každé podstatné meno do jedného z definovaných synsetov. Jednotlivé rysy, ktorými autor reprezentoval získané informácie, potom vyjadrujú odpoveď na otázky, či sa vo vete vyskytuje podstatné meno, ktoré patrí do zadaného synsetu.

V ďalšej časti autor prezentuje získané výsledky. Experimetoval nie len s rôznymi metódami strojového učenia, ale tiež s rôznymi kombináciami druhov rysov (popísaných vyššie). Na základe týchto informácií vyslovuje záver, že najdôležitejším druhom rysov, ktoré najviac prispievajú k úspešnosti klasifikátorov sú syntaktické rysy.

Autor experimentoval na dvoch rôznych sadách údajov. Na menšej sade (VALEVAL) bola najúspešnejšia metóda rozhodovacích stromov (konkrétne algoritmus C5). Na väčšej sade získanej z PDT boli najúspešnejšie metóda podporných vektorov (SVM) a metóda maximálnej entropie.

Ako baseline si autor zvolil klasifikátor, ktorý každej inštancii priradí najpravdepodobnejší valenčný rámec – tj. ten, ktorý sa v trénovacíh údajoch vyskytoval najčastejšie. V porovaní s baseline dosiahli metódy strojového zlepšenia štatisticky signifikatné zlepšenie – 12% na sade údajov z VALEVAL a 6,5% na sade údajov z PDT.

Jednou z možností na zlepšenie, ktorú autor uvádza v závere práce, je spôsob, akým by sa dali zväčšiť trénovacie údaje. Navrhuje zlúčiť valenčné rámce, ktoré majú podobné chovanie a ktoré metódy nedokážu spoľahlivo rozlišovať.

4.3 Pravidlový klasifikátor patternov

V diplomovej práci [14] je popísaný návrh, implementácia a evaluácia prvého pokusu o automatický klasifikátor patternov slovníka PDEV. Na tomto mieste uvádzame rešerš práce.

Prvým krokom k implementácii pravidlového klasifikátora bolo vytvoriť efektívny postup na to, ako extrahovať potrebné argumenty slovesa zo zadanej vety. Jedná sa predovšetkým o argumenty, ktoré popisujú jednotlivé definície patternov v PDEV – subjekt a objekt (priamy aj nepriamy).

V práci [14] sa tieto argumenty extrahujú z tzv. stanfordských závislostí[1] (*Stanford dependencies*). Tie môžu byť rôznych typov a je ich možné získať pomocou viacerých parserov (priamo, alebo následnou úpravou výstupného formátu).

V práci sú popísané experimenty s niekoľkými syntaktickými analyzátormi. Jedná sa o Stanfordský parser [41], Charniak-Johnsonov parser [42] a závislostný parser od McDonalda [43] (v rámci aplikácie TectoMT [44]). Vo všetkých prípadoch existujú procedúry, ktoré dokážu previesť výstup týchto parserov do formátu stanfordských závislostí alebo majú priamo výstup v tomto formáte. Ďalšie experimenty sú popísané aj s použitím pravidlového parsera v aplikácii Sketch Engine [45], ktorý extrahoval subjekty a objekty priamo.

Dôležitou súčasťou práce je experimentálne vyhodnotenie úspešnosti jednotlivých parserov z hľadiska toho, ako úspešne je možné ich výstupy použiť pri získavaní argumentov slovies v zadaných vetách. Ako evaluačnú mieru pre výsledky experimentov autorka zvolila F-measure[2]. Najlepším parserom sa stal Charniak-Johnsonov parser. Stanfordský parser bol horší len o 1,5%. Keďže však je jeho použitie najjednoduchšie a nie sú potrebné žiadne ďalšie prevody z jeho výstupného formátu na stanfordské závislosti, v práci sa ďalej experimentuje už len s týmto parserom.

Asi najčastejším spôsobom, akým sú argumenty u jednotlivých patternov definované, je použitie sémantických typov. Správne priradzovanie slov ku sémantickým typom je kľúčovou vlastnosťou pre automatický klasifikátor. V práci je táto úloha riešená vytvorením zoznamov slov, ktoré patria k jednotlivým sémantickým typom (vytvára sa *populácia sémantických typov*).

Autorka použila niekoľko nástrojov, aby popisované populácie slov získala. Predovšetkým využila anotované vety zo slovníku PDEV a extrakciu argumentov z týchto viet. Okrem toho sa ako veľmi užitočný nástroj prejavil rozpoznávač menných entít (NER, z anglického *Name Entity Recognizer*) [46], ktorý v dodanej vete dokáže označit vlastné mená osôb, organizácií a geografické názvy. Ďalším problémom, ktorý bol v práci popísaný a riešený, bol prevod množného čísla substantív na jednotné. Vyriešený bol vytvorením slovníka, ktorý obsahoval tvary substantív v množnom a jednotnom čísle a umožňoval tak prevod medzi nimi.

Samotné zaradzovanie slov do zoznamov realizujúcich jednotlivé sémantické typy nie je triviálnou úlohou. V prípade, že argument patternu je definovaný pomocou viacerých

[1] Podrobnosti o stanfordských závislostiach uvádzame v dodatku B.

[2] Túto mieru definujeme v časti 3.4 tejto práce.

sémantických typov, napríklad [[**Human** — **Institution**]], nevieme, do ktorého zo zoznamov ho zaradiť a musíme si pomôcť inak. Napríklad tak, že ak máme k dispozícii niekoľko výskytov tohto slova, môžeme pomocou rôznych štatistických mier určiť, do ktorého z týchto sémantických typov patrí slovo viac a do ktorého menej. V práci sú popísané 4 rôzne miery – F_1, F_2, F_3, F_4. V ich definíciách sa používa funkcia $c(S, w)$, ktorá vyjadruje počet inštancií, v ktorých je slovo w na mieste argumentu definovaného množinou sémantických typov S.

$$F_1(t, w) = \sum_{t \in S} c(S, w)$$

$$F_2(t, w) = c(\{t\}, w)$$

$$F_3(t, w) = \sum_{t \in S} \frac{1}{|S|} c(S, w)$$

$$F_4(t, w) = \sum_{t \in S} \frac{c(\{t\}, w)}{\sum_{|S|=1} c(S, w)} c(S, w)$$

Význam jednotlivých mier uvádzame v tomto prehľade:

- $F_1(t, w)$ vyjadruje počet inštancií, v ktorých sa slovo w vyskytlo v súvislosti so sémantickým typom t
- $F_2(t, w)$ vyjadruje počet inštancií, v ktorých sa slovo w vyskytlo samostatne so sémantickým typom t
- $F_3(t, w)$ vyjadruje počet inštancií, ktoré pripadnú na jeden sémantický typ t, ak stanovíme, že celkový počet výskytov slova sa delí rovnomerne na sémantické typy, s ktorými sa vyskytol vo dvojici (S, w).
- $F_4(t, w)$ vyjadruje tzv. prírastok atribútu F_4. Dáva do pomeru počty samostatných výskytov slova w so sémantickým typom t a ostatnými sémantickými typmi, s ktorými sa slovo w vyskytlo samostatne.

Okrem týchto mier sú v práci definované aj ďalšie 2 miery – PMI_3 a PMI_4, odvodené z mier F_3 a F_4 nasledujúcim spôsobom:

$$PMI_3(t, w) = \log_2 \frac{N \cdot F_3(t, w)}{\sum_{w \in W} F_3(t, w) \cdot \sum_{w \in T} F_3(t, w)}$$

$$PMI_4(t, w) = \log_2 \frac{N \cdot F_4(t, w)}{\sum_{w \in W} F_4(t, w) \cdot \sum_{w \in T} F_4(t, w)}$$

Jedná sa o tzv. bodovú zvájomnú informáciu (*pointwise mutual information*), kde N je počet všetkých dvojíc (S, w), W je slovná zásoba a T je zoznam všetkých sémenatických typov.

Na základe týchto mier bol každému spracovávanému substantívu priradený jeden (najpravdepodobnejší) sémantický typ. Sémantické typy navrhol Patrick Hanks intuitívne a v čase písania práce nebol k dispozícii žiaden experiment, ktorý by overil, či zaradzovanie slov do sémantických typov je úloha s dobrou IAA. V práci je popísaný experiment, v ktorom bolo náhodne vybraných 3000 slov a sémantických typov, ktoré k nim boli priradené a Patrick Hanks mal určiť, či je priradenie správne alebo nie. Výsledky uvádzame v tabuľke 4.1.

Značka	Počet	
T	1139	38,1%
C	483	16,2%
M	1367	45,7%

Tabuľka 4.1: Výsledky experimentu s automatickým generovaním populácie sémantických typov.

V poslednej časti práce je popísaný samotný pravidlový klasifikátor (bez metód strojového učenia), ktorý na základe populácií sémantických typov získaných v predchádzajúcej časti priradzuje jednotlivým vetám patterny. V práci je navrhnutý efektívny postup, akým zlepšovať kvalitu klasifikátora a tiež kvalitu populácie sémantických typov:

- Na základe existujúcej populácie sémantických typov sa vytvorí klasifikátor, ktorým budeme môcť získať ďalšie anotované vety.
- Anotované vety sú základným zdrojom informácií pre vytváranie populácií sémantických typov.
- Pomocou vylepšeného populačného zoznamu by klasifikátor mal dosiahnut lepšiu úspešnosť a mohol anotovať ďalšie vety.

Pri evaluácií klasifikátora bolo použitých približne 700 v tej dobe skompilovaných slovies, ktoré dokopy tvorili zbierku asi 200000 slovesných tokenov. Trénovacie a testovacie údaje boli rozdelené v pomere 2:1. K vytvoreniu populácie sémantických typov boli však použité i testovacie dáta a to kvôli tomu, aby populácie sémantických typov mali čo najväčšiu veľkosť.

V čase písania práce neboli v projekte PDEV definované exploatácie `.a`, `.c`, `.f` a `.s`. Namiesto nich anotátori používali značku `.e` pre všetky prípady, ktoré pokrývajú exploatácie v súčasnosti. Pri evaluácii sa nepovažovalo za chybu, ak klasifikátor priradil k patternu značku `.e`, alebo značku `.e` nepriradil.

Vstupom klasifikátora boli nasledujúce zdroje:

- populácie sémantických typov
- definície patternov (vo formáte CSV)
- slovník na prevod singuláru a plurálu
- súbor pravdepodobností priradenia patternu pre dané sloveso
- vety s vyznačeným slovesným tokenom
- stanfordské závislosti pre jednotlivé vety
- výstup aplikácie NER

Úspešnosť klasifikátora bola meraná pomocou miery Accuracy[3]. Celková úspešnosť klasifikátora bola vyjadrená ako vážený priemer úspešností nad jednotlivými slovesami. Váhou vo váženom priemere bol podiel pokrytia slovesa v BNC50. Väčší dôraz sa teda kládol na slovesá, ktoré pokrývajú viac viet. Úspešnosť klasifikátora bola 72,86%.

[3]Mieru Accuracy podrobne popisujeme a definujeme v časti 3.4 tejto práce.

Kapitola 5

Analýza dostupných údajov

V tejto kapitole sa podrobne zaoberáme údajmi, s ktorými pracujeme v tejto práci. Základné jednotky – konkordancie, z ktorých budeme tvoriť vektory rysov, predstavujú konkordancie v anglickom jazyku pre jedno z 30 cieľových slovies, pre ktoré sa v tejto práci snažíme vytvoriť čo možno najúspešnejší klasifikátor patternov.

Čitateľovi popíšeme formát vstupných údajov, základné štatistiky, akými sú počet konkordancií, frekvencie cieľových slovies v korpuse a pod. Ďalej v tejto kapitole popíšeme spôsob, akým boli jednotlivé konkordancie anotované a akým bola vyhodnocovaná medzianotátorská zhoda. Budeme sa zaoberať frekvenciou jednotlivých patternov a na koniec definujeme množinu výstupných tried, s ktorými bude klasifikátor pracovať.

5.1 Základné štatistiky kolekcie VPS

Základné charakteristiky 30 cieľových slovies, s ktorými sa v práci zaoberáme, sú uvedené v tabuľke 5.1. Medzi základné charakteristiky sme zaradili nasledujúce veličiny:

- počet konkordancií pre jednotlivé slovesá,
- frekvencie slovies v korpuse.

Na ich základe sme vytvorili rozdelenie slovies do skupín podľa frekvencie výskytu, ktoré používame na reprezentatívnejšie hodnotenie výkonnosti modelov.

5.1.1 Počet konkordancií pre jednotlivé slovesá

Pre každé sloveso máme k dizpozícii rôzny počet anotovaných konkordancií, ktoré môžeme použiť na testovanie a trénovanie klasifikátorov.

Vo väčšine prípadov je pre každé sloveso k dizpocícii tzv. *referenčná vzorka*, ktorá vo väčšine prípadov obsahuje 300 konkordancií. Každú konkordanciu anotoval lexikograf jednou značkou označujúcou pattern. Referenčnú vzorku mali k dizpozícii anotátori, ktorí anotovali ďalšie konkordancie.

Pre každé sloveso je tiež k dizpocícii multianotovaná vzorka 50 konkordancií, ktoré anotovali 4 nezávislí anotátori. Anotovanie štyroch anotátorov prebiehalo väčšinou v niekoľkých kolách. Anotácie z predchádzajúcich kôl sa pripojili k referenčnej vzorke, anotácie konkordancií v poslednom kole sa stali základom pre určovanie medzianotátorskej zhody. Viac o meraní medzianotátorskej zhody uvádzame v sekcii 2.3.3.

V tabuľke 5.2 môžeme vidieť, že pre väčšinu cieľových slovies máme k dizpocícii 300-350 konkordancií. Pre väčšinu slovies máme tiež k dizpozícii 50 multianotovaných konkordancií. Pre dve slovesá (*plug* a *enlist*) máme k dizpozícii len 46, resp. 47 multianotovaných konkordancií. Je to spôsobené malou frekvenciou výskytu týchto slovies

	Počet konkordancií			Frekvencia v korpuse			Skupiny	
Sloveso	**Referenčná vzorka**	**Multianotovaná vzorka**	**Počet celkom**	**Počet konkordancií**	**Podiel konkordancií**	**Váha slovesa**	**Skupina**	**Váha v skupine**
access	300	50	350	455	0.010%	0.29%	C	4.04%
ally	250	50	300	386	0.008%	0.24%	C	3.42%
arrive	250	50	300	6110	0.131%	3.83%	B	31.22%
breathe	350	50	400	950	0.020%	0.60%	C	8.43%
claim	500	50	550	12517	0.268%	7.85%	A	9.73%
cool	300	50	350	575	0.012%	0.36%	C	5.10%
crush	350	50	400	435	0.009%	0.27%	C	3.86%
cry	250	50	300	1200	0.026%	0.75%	B	6.13%
deny	300	50	350	4811	0.103%	3.02%	B	24.58%
enlarge	300	50	350	529	0.011%	0.33%	C	4.69%
enlist	300	47	347	347	0.007%	0.22%	C	3.08%
forge	350	50	400	511	0.011%	0.32%	C	4.53%
furnish	300	50	350	396	0.008%	0.25%	C	3.51%
hail	300	50	350	775	0.017%	0.49%	C	6.88%
halt	250	50	300	856	0.018%	0.54%	C	7.59%
part	300	50	350	380	0.008%	0.24%	C	3.37%
plough	250	50	300	357	0.008%	0.22%	C	3.17%
plug	300	46	346	346	0.007%	0.22%	C	3.07%
pour	300	50	350	914	0.020%	0.57%	C	8.11%
say	500	50	550	94608	2.025%	59.31%	A	73.52%
smash	300	50	350	530	0.011%	0.33%	C	4.70%
smell	300	50	350	416	0.009%	0.26%	C	3.69%
steer	300	50	350	432	0.009%	0.27%	C	3.83%
submit	250	50	300	2258	0.048%	1.42%	B	11.54%
swell	300	50	350	398	0.009%	0.25%	C	3.53%
tell	500	50	550	21550	0.461%	13.51%	A	16.75%
throw	1000	50	1050	3710	0.079%	2.33%	B	18.96%
trouble	300	50	350	375	0.008%	0.24%	C	3.33%
wake	300	50	350	909	0.019%	0.57%	C	8.06%
yield	300	50	350	1482	0.032%	0.93%	B	7.57%
Celkom	10150	1493	11643	159518	3.414%	100.00%		300.00%

Tabuľka 5.1: Základné charakteristiky 30 cieľových slovies

v korpuse. Po vytvorení referenčnej vzorky a prvého kola anotácií už v korpuse zostalo menej ako 50 neanotovaných konkordancií.

Špeciálne je tiež sloveso throw, pre ktoré máme k dizpocícii až 1050 konkordancií. Pre toto sloveso bola zvolená výrazne väčšia referenčná vzorka, pretože sa jedná o veľmi komplexné sloveso.

Počet slovies	**Počet konkordancií**
15	350
6	300
3	550
3	400
1	347
1	346
1	1050

Tabuľka 5.2: Distribúcia počtu konkordancií pre cieľové slovesá

5.1.2 Frekvencie slovies v korpuse

Všetky konkordancie v slovníku PDEV boli získané z Britského národného korpusu[1] (BNC) [18]. Pre každé zo slovies v slovníku PDEV preto môžeme určiť frekvenciu výskytu slovesa ako počet konkordancií v korpuse.

Na základe frekvencie slovesa môžme vypočítať jeho pokrytie v korpuse ako podiel počtu konkordancií obsahujúcich dané sloveso a počtu všetkých konkordancií. Ako počet všetkých konkordancií rozumieme počet viet, v ktorých sa vyskytuje nejaké sloveso, ktoré zároveň nie je slovesom *be*, *have* alebo modálnym slovesom. Takýchto konkordancií je v BNC50 4673093. V poslednom riadku tabuľky 5.1, v stĺpci Podiel konkordancií uvádzame, že celkové pokrytie korpusu slovesami, s ktorými pracujeme je $3,41\%$.

Výber slovies, ktoré sa dostali do kolekcie 30 slovies, s ktorými pracujeme, sa uskutočnil so zámerom vybrať reprezentatívnu vzorku slovies z korpusu. V kolekcii sa vyskytujú veľmi frekventované slovesá (*say*, *tell*) ale aj veľmi ojedinelé slovesá (*plug*, *enlist*), preto je celkový súčet frekvencií slovies relatívne malý.

5.1.3 Rozdelenie slovies do skupín

Hoci v práci vytvárame nezávislé modely pre každé z 30 slovies, môžeme sa na úlohu priradzovania patternov pozerať aj ako na čiernu skrinku, ktorá priradzuje patterny náhodne zvoleným konkordanciám obsahujúcim jedno z cieľových slovies.

Rozhodli sme sa vyjadriť celkovú úspečnosť všetkých klasifikátorov jedným číslom ako úspešnosť takejto čiernej skrinky. Úspešnosť by sme mohli definovať viacerými spôsobmi, napríklad ako priemer úspešnosti jednotlivých modelov. Ak by sme do skrinky vkladali náhodné konkordancie, skrinka by musela častejšie klasifikovať slovesá, ktoré sa v korpuse vyskytujú s vyššou frekvenciou než slovesá vyskytujúce sa s malou frekvenciou. Preto vyjadrenie úspešnosti ako prostý priemer by neodpovedalo úspešnosti, akú by čierna skrinka dosiahla, ale blížila by sa úspečnosti klasifikátora pre najčastejšie hodnotené sloveso – najčastejšie sloveso v kolekcii.

Na základe tejto úvahy sme sa rozhodli vyjadriť úspešnosť ako vážený priemer úspešnosti jednotlivých klasifikátorov. Váhou pre každé sloveso je podiel tohto slovesa

[1] Podrobnosti o BNC uvádzame v sekcii 2.1.

na pokrytí konkordancií obsahujúcich našich 30 cieľových slovies. Táto váha je pre každé sloveso vyčíslená v stĺpci Váha slovesa v tabuľke 5.1.

Z uvedených štatistík vyplýva, že slovesá, s ktorými pracujeme, majú veľmi rôzne frekvencie a tým aj váhy. V celej kolekcii dominuje sloveso *say*, ktoré tvorí 59, 31% zo všetkých konkordancií obsahujúcich jedno z nami spracovávaných slovies. Vo váženom priemere, ktorý sme definovali v predchádzajúcom odstavci, by potom bola úspešnosť klasifikátora pre sloveso *say* najdôležitejšia a údaj o úspešnosti by nebol dostatočne reprezentatívny.

Rozhodli sme sa preto rozdeliť slová do troch skupín – A, B a C. Každá skupina združuje slovesá s približne rovnakou frekvenciou. Skupina A obsahuje tri slovesá, frekvencia každého z nich je aspoň 10000 konkordancií. Skupina B obsahuje 6 slovies, ktorých frekvencia v BNC50 je od 1000 do 10000 konkordancií. Do skupiny C sme zaradili zvyšné slovesá, ktorých frekvencia v BNC50 je menšia než 1000 konkordancií. Pre každú z týchto skupín budeme určovať úspešnosť klasifikátorov ako vážený priemer úspešnosti v danej skupine. Na záver môžme určiť aj celkovú úspešnosť všetkých klasifikátorov ako vážený priemer úspešností týchto troch skupín. Zaradenie slovies do skupín je uvedené v stĺpci skupina v tabuľke 5.1.

Uvádzame aj prehľadnú tabuľku 5.1.3 s charakteristikou skupín A, B a C.

Skupina	**Počet slovies**	**Počet konkordancií**	**Váha skupiny**
A	3	128675	80.66%
B	6	19571	12.27%
C	21	11272	7.07%
Celkom	30	159518	100.00%

Tabuľka 5.3: Základné charakteristiky troch frekvenčných skupín

5.2 Distribúcia patternov

Nedeliteľnou súčasťou kolekcie VPS je okrem definícií patternov pre jednotlivé slovesá aj zbierka anotovaných konkordancií z BNC50. Pre väčšinu konkordancií v slovníku je k dizpozícii jedna značka, ktorú konkordancii priradil anotátor – skúsený lexikograf.

Sémantické značky, ktoré anotátori priradzovali jednotlivým konkordanciám, sme popísali v časti 2.1.2. Najčastejšími značkami v kolekcii sú značky vyjadrené číslami, napr. 1, 2, 3, ktoré reprezentujú dané paterny. Okrem nich mohli anotátori ku každej takejto značke priradiť ešte jedno z písem `a`, `c`, `f` a `s`, ktoré indikujú exploatácie. V prípade, že sa v kolekcii vyskytla konkordancia bez cieľového slovesa (vďaka chybe pri prípave vzoriek), anotátori použili značku `x`. Poslednou značkou, ktorú mohli anotátori použiť, bola značka `u` pre konkordancie, ktorých význam nebol zachytený ani v jednom z definovaných patternov.

Perplexita patternov

Pre každé sloveso z kolekcie je definovaný rôzny počet patternov. Každý pattern predstavuje určitý vzor najčastejšieho použitia daného slovesa. Slovesá sa výrazne líšia počtom patternov. Existujú slovesá, ktoré majú 4 patterny a existujú slovesá, ktoré majú desiatky patternov.

Hoci sa jedná o najčastejšie použitia, každý pattern môže mať navyše rôznu frekvenciu. Frekvencie jednotlivých patternov sú pre každé sloveso rôzne. Existujú slovesá, v ktorých dominuje jeden pattern a niekoľko ďalších má malé frekvencie, alebo,

na druhej strane, existujú slovesá, ktoré majú niekoľko patternov, ktoré majú približne rovnakú frekvenciu.

Na vyjadrenie rôznych počtov patternov a rôznych frekvencií patternov jedným číslom použijeme miery z teórie informácie – entropiu a perplexitu.

Keďže anotované konkordancie, ktoré máme k dizpocícii, boli z korpusu vyberané náhodne, môžeme predpokladať, že frekvencia patternov v anotovaných konkordanciách približne zodpovedá frekvencii patternov v celom korpuse (dokonca v celom jazyku, ak predpokladáme, že konkordancie v korpuse sú reprezentatívnou vzorkou anglického jazyka).

Na základe tejto úvahy môžme o frekvenci patternov uvažovať ako o distribúcii náhodnej premennej X. Pre každé sloveso bude mať náhodná premenná X definované všetky patterny ako hodnoty, ktoré môže nadobúdať. Pre náhodnú premennú definujeme pravdepodobnosť $p(x)$, ktorá vyjadruje, že náhodná premenná X nadobúda hodnotu x práve s pravdepodobnosťou $p(x)$. Túto pravdepodobnosť môžme vypočítať podľa predpisu

$$p(x) = \frac{c(x)}{|I|},$$

kde $c(x)$ je frekvencia výskytu patternu x v množine konkordancií I.

Entropia náhodnej premennej X je potom daná predpisom

$$H(X) = -\sum_{x \in X} \log_2(p(x)) \cdot p(x).$$

Perplexita náhodnej premennej X sa spočíta z entropie predpisom

$$P(X) = 2^{H(X)}$$

Nízka hodnota perplexity indikuje, že sloveso má malý počet patternov, alebo sú patterny rozdelené nerovnomerne (napríklad existuje dominantný pattern, ktorý má oveľa vyššiu frekvenciu ako všetky ostatné). Maximálnu hodnotu dosiahne perplexita v prípade, že sloveso má veľký počet patternov a všetky majú približne rovnaké frekvencie výskytov.

Zlučovanie exploatácií

V ideálnom prípade by modely klasifikátorov mali klasifikovať všetky možné sémantické značky. Trénovanie takýchto klasifikátorov by ale veľmi rýchlo stroskotalo na probléme s nedostatkom údajov. Vo všeobecnosti platí, že model potrebuje aspoň 5-10 konkordancií na trénovanie (a minimálne 1 konkordanciu na testovanie). Pri sledovaní distribúcie patternov v kolekcii sme zistili, že podmienku na dostatočný počet údajov nespĺňa viacero patternov.

V snahe zvýšiť hustotu dát (hustotu môžme intuitívne chápať ako priemerný počet konkordancií na pattern) sme sa rozhodli ignorovať exploatácie a pripojiť konkordancie anotované exploatačnými značkami k hlavným patternom. Existujú minimálne 2 spôsoby, akými sme spojenie mohli realizovať:

1. **Sémantické zlučovanie**. Pri sémenaticky motivovanom zlučovaní značiek by sme zlúčili exploatácie `a`, `c` a `s` s hlavným patternom a exploatáciu `f` so značkou `u`. Exploatácie `a`, `c` a `s` totiž informujú len o formálnej výnimke oproti definícii patternu, pričom význam konkordancie ostáva zhodný s definíciou patternu. Naopak, exploatácia `f` označuje figuratívne použitie – syntakticky i morfologicky konkordancia vyhovuje definícii patternu, ale význam konkordancie nezodpovedá významu patternu. Zaradili by sme ju preto do značky `u`, v ktorej skladujeme konkordancie, pre ktoré nie je definovaný pattern a to najmä pre ich nízku frekvenciu.

2. **Zanedbanie exploatácií**. Princíp tohto zlučovania spočíva v jednoduchom strojovom spracovaní exploatačných značiek. Exploatácie sú jednoduchou úpravou odstránené, tj. všetky exploatácie `a`, `c`, `f` a `s` sú spojené s hlavným patternom.

V práci sme zvolili druhú z popisovaných možností. Rozhodli sme sa pre ňu najmä kvôli jej jednoduchosti a tiež preto, že v celej kolekcii neexistuje mnoho konkordancií so značkou `f`.

Myslíme si, že spojením exploatácií s normálnymi použitiami patternov neprichádzame o veľké množstvo informácie a spojenie nám v niektorých prípadoch umožní klasifikovať patterny, na ktoré by sme v opačnom prípade nemali dostatok trénovacích konkordancií.

Od tejto chvíle budeme sémantické značky stotožňovať s patternami a zároveň tým budeme mať na mysli, že sme všetky exploatácie spojili s hlavnými patternami. Pre každé sloveso nám teda v množine sémantických značiek ostali len značky pre patterny (čísla) a značky `x` a `u`, ak sa také vyskytli medzi konkordanciami. V ďalšom texte budeme pre jednoduchosť za patterny považovať aj značky `x` a `u` a tým môžme považovať slová patterny a sémantické značky, ktorými budú klasifikátory klasifikovať, za synonymné.

Voľba prahovej frekvencie patternov

Aj napriek spojeniu exploatácií s hlavnými patternami v kolekcii existovali patterny, ktoré mali veľmi malú frekvenciu. Algoritmy strojového učenia by sa tieto patterny nemali šancu naučiť a preto sme sa rozhodli v ďalších experimentoch uvažovať len s patternami s dostatočne vysokým počtom konkordancií. Hranica v počte konkordancií, ktorú museli patterny prekročiť, sa nazýva prahová frekvencia (*threshold*).

Voľbe prahovej frekvencie sme venovali primerané množstvo času a realizovali sme niekoľko experimentov v snahe určiť jej optimálnu hodnotu. Experimenty a ich praktické závery popisujeme v časti 7.2. Po týchto experimentoch sme sa rozhodli, že prahovú frekvenciu nastavíme tak, aby sa klasifikátory učili klasifikovať len patterny, ktoré majú aspoň 9 konkordancií.

Konkordancie, ktoré boli klasifikované patternami s nižšou frekvenciou, sme z kolekcie nevyhodili, ale zmenili sme im ich sémenatickú značku na značku `u`. Motiváciou nám bola myšlienka, že zachovaním konkordancií v trénovacích údajoch dávame algoritmom možnosť učiť sa aj na negatívnych prípadoch, tj. že konkordancia nepatrí do daného patternu. Vylúčením konkordancií z trénovacej množiny by sme sa o túto možnosť pripavili.

Spojenie málo frekventovaných patternov do značky `u` je zároveň úplne v súlade s metodológiou PDEV a CPA, ktorá prepodkladá definovanie patternov len pre najčastejšie použitia slovies. Nastavením prahovej frekvencie sme jednoducho zvýšili hranicu toho, čo považujeme za dostatočné množstvo konkordancií na to, aby sa pre ne definoval vlastný pattern.

Tabuľka 5.2 zobrazuje niekoľko charakteristík pre jednotlivé slovesá pred a po použití prahovej frekvencie. V každom z uvedených stĺpcov uvádzame priemerný počet sledovanej veličiny pre danú skupinu a celkovo pre celú kolekciu.

5.3 Medzianotátorská zhoda

Pre úplnosť v tejto časti prezentujeme aj štatistiky medzinanotátorskej zhody kolekcie VPS. Po zlúčení exploatácií a aplikovaní prahovej frekvencie sa zmenili i miery IAA – Fleissova kapa a ARG.

Tabuľka 5.3 porovnáva výsledky merania medzianotátorskej zhody pomocou Fleissovej kapy a pomocou miery ARG pred a po aplikácií prahovej frekvencie. V jed-

			Pred prah. frek.			Po prah. frek.		
Sloveso	**Váha v skupine**	**Počet konkordancií**	**Počet patternov**	**Priemerný počet konkordancií na pattern**	**Perplexita**	**Počet patternov**	**Priemerný počet konkordancií na pattern**	**Perplexita**
A	80.66%	500	16	32	1.870	7	79	2.320
B	12.27%	408	22	28	8.949	10	42	6.482
C	7.07%	301	13	28	6.850	7	45	5.342
Priemer celkom		475	16	31	3.090	7	72	3.044

Tabuľka 5.4: Zmena vybraných charakteristík slovies po aplikovaní prahovej frekvencie. Podrobnú tabuľku pre všetky slovesá uvádzame v dodatku D.2.

notlivých stĺcoch sú uvedené priemerné hodnoty Fleissovej kapy a ARG pre jednotlivé skupiny a celkovo pre celú kolekciu.

			Pred prah. frek.		Po prah. frek.	
Sloveso	**Skupina**	**Váha v skupine**	**Fleiss kappa**	**ARG**	**Fleiss kappa**	**ARG**
A		80.66%	0.879	0.947	0.902	1.035
B		12.27%	0.782	1.122	0.788	1.197
C		7.07%	0.782	1.122	0.788	1.197
Priemer celkom			0.860	0.981	0.880	1.067

Tabuľka 5.5: Porovanie dvoch mier medzianotátorskej zhody pred a po aplikácii prahovej frekvencie. Podrobné výsledky pre všetky slovesá uvádzame v dodatku D.3

5.4 Formát vstupných údajov

Anotované konkordancie

Asi najdôležitejšou časťou vstupných údajov je kolekcia anotovaných viet z BNC50, ktorá predstavuje referečnú vzorku. Pre každé sloveso existuje jednoduchý textový súbor, ktorého tvar je možné vidieť na obrázku 5.4. Okrem samotných konkordancií súbor obsahuje aj frekvenčný zoznam sémantických značiek použitých v súbore.

Po frekvenčnom zozname nasledujú jednotlivé konkordancie. Každá konkordancia je zapísaná na samostatnom riadku. Riadok začína znakom a pokračuje identifikačným číslom konkordancie. Toto identifikačné číslo predstavuje smerník do BNC50, ktorý je

```
access-goldrs1:   50
        1         20
        1.s        1
        2          2
        2.f        1
        3          2
        3.a        1
#2232067 [1] Many PD sources offer Hypercard ‘ stacks ’ , which ...
#8058916 [u] Since distribution concerns <access> the requirements ...
#8771826 [x] Thirty-six hours from the relevant time would expire ...
#9478137 [1] For the purposes of stock revision , back files of ...
```

Obr. 5.1: Ukážka formátu vstupného súboru obsahujúceho jednu anotáciu pre každú konkordanciu.

#agrs	AL	JT	SC	EK	multi	adjudication	sent_id	sent
0	x	2.a	2	1.a	1.a, 2, 2.a, x	2.a	40095711	...
1	x	1.a	1	1	1, 1.a, x	1.a	11463134	...
1	u	5.a	u	1.a	1.a, 5.a, u	u	20684105	....
1	x	x	u	3.f	3.f, u, x	u	21660073	...
1	4	3	4.a	4	4, 4.a	4.a	27316591	...

Obr. 5.2: Ukážka formátu vstupného súboru obsahujúceho multianotácie.

uložený na serveri projektu PDEV. Za identifikáciou konkordancie nasleduje v hranatých zátvorkách sémantická značka a za ňou samotná veta. Cieľové sloveso je navyše ohraničené v špicatých zátvorkách.

Multianotácie

Adjudikovaná vzorka 50 konkordancií, ktoré anotovali súčasne 4 anotátori je uložená pre každé sloveso v jednom CSV súbore. Okrem číselnej identifikácie konkordancie a textu anotovanej vety obsahuje aj anotáciu každého zo 4 anotátorov (stĺpce sú označené iniciálami mien anotátorov). Pre rýchlu orientáciu je v súbore stĺpec #args, ktorý obsahuje počet zhôd medzi dvoma anotátormi. Maximálna hodnota pre 4 anotátorov je teda 6. V stĺpci multi je uvedený zoznam sémantických značiek, ktoré lexikograf vyhodnotil ako prípustné pre danú konkordanciu. Adjudikovaná značka, ktorá predstavuje goldstandard je nakoniec uvedená v stĺpci adjudication.

Ukážka súboru je na obrázku 5.4

Matice konfúzie

Nech T je množina možných sémantických značiek indexovaná číslami $i = 1, ..., |T|$ a $j = 1, ..., |T|$. Matica konfúzie C je matica, ktorú môžeme definovať pre každú dvojicu anotátorov. Na pozícii C_{ij} obsahuje počet konkordancií, ktoré prvý anotátor anotoval tagom i a druhý anotátor tagom j.

Je zrejmé, že na diagonále bude táto matica obsahovať počet zhôd oboch anotátorov na jednotlivých tagoch. Kolekcia VPS obsahuje matice konfúzií pre každé sloveso v samostatnom textovom súbore. V každom súbore nájdeme matice pre každú možnú dvojicu anotátorov, ktorí anotovali vzorku 50 konkordancií, ktoré sú uložené v multianotačných súboroch.

```
+----------------------+
| Annotators: AV vs. EK |
+----------------------+

      |    1  1.a  1.s    2    3  3.f    4    u    x |    #  prob
------+-----------------------------------------------+------------
    1 |   24    -    -    1    -    -    -    -    - |   25  0.50
  1.a |    -    -    -    -    -    -    -    -    - |    0  0.00
  1.s |    -    -    1    -    -    -    -    -    - |    1  0.02
    2 |    -    -    -    2    -    -    -    -    - |    2  0.04
    3 |    -    -    -    -    1    -    -    -    1 |    2  0.04
  3.f |    -    1    -    -    -    -    -    -    - |    1  0.02
    4 |    -    -    -    -    -    -    2    -    - |    2  0.04
    u |    -    1    -    -    -    -    -    -    - |    1  0.02
    x |    3    1    -    -    -    1    -    -   11 |   16  0.32
------+-----------------------------------------------+------------
    # |   27    3    1    3    1    1    2    0   12 |   50
 prob | 0.54 0.06 0.02 0.06 0.02 0.02 0.04 0.00 0.24 |       1.00
```

Obr. 5.3: Ukážka formátu vstupného súboru obsahujúceho matice konfúzie.

Súbory s maticami konfúzií slúžia ako vstup aplikácie na výpočet ARG. Ukážku súboru uvádzame na obrázku 5.4.

Výstup morfologickej analýzy

Každá konkordancia z kolekcie VPS bola spracovaná morfologickou analýzou, ktorá ku každému slovu priradila jeho lemmu a morfologickú značku. Morfologické značky, ktoré priradzovala morfologická analýza sú súčasťou tagsetu korpusu Penn Tree Bank [47] a ich prehľad uvádzame v dodatku A. Vstupný súbor s výstupom morfologickej analýzy obsahoval na každom riadku jednu spracovanú konkordanciu. Za každým slovom nasledovala lemma a morfologický tag, oddelené tabulátorom.

Výstup syntaktickej analýzy

Každá konkordancia VPS bola spracovaná Stanfordským parserom [41], ktorý vytvoril syntaktickú analýzu konkordancie v podobe stanfordských závislostí. Výstupom analýzy je súbor, v ktorom je uvedená každá závislosť na samostatnom riadku. Jednotlivé konkordancie sú oddelené prázdnym riadkom.

Popis a stručný prehľad stanfordských závislostí uvádzame v dodatku B.

Výstup analýzy menných entít

Každá konkordancia v kolekcii VPS bola spracovaná Stanfordským rozpoznávačom menných entít (NER) [46], ktorý v konkordanciách identifikuje vlastné mená osôb, organizácií a zemepisných reálií. Výstupom aplikácie NER je súbor, v ktorom je na každom riadku analyzovaná jedna konkordancia. Ku každému slovu je za znakom `/` uvedená jedna z nasledujúcich značiek:

- `PERSON` pre mená osôb,
- `ORGANIZATION` pre názvy organizácií a inštitúcií,
- `LOCATION` pre názvy geografických reálií a
- `O` pre ostatné slová

Kapitola 6

Návrh rysov pre strojové učenie

V tejto kapitole sa zaoberáme otázkou, akými rysmi je možné čo najlepšie reprezentovať morfologické, syntaktické a sémantické informácie o danej konkordancii, ktoré by boli vhodné pre algoritmy strojového učenia. Navrhnuté rysy sme rozdelili na dve hlavné skupiny – morfologicko-syntaktické a sémantické rysy.

6.1 Morfologicko-syntaktické rysy

Martin Holub [48] v spolupráci so Silviou Cinkovou navrhli sadu morfologických a syntaktických rysov. Táto množina rysov sa snaží zachytiť všetky dôležité morfologické a syntaktické aspekty kontextu slovesa. Rysy v nej obsiahnuté môžeme rozdeliť do niekoľkých skupín:

1. Rysy charakterizujúce cieľové sloveso
2. Rysy charakterizujúce najbližší kontext cieľového slovesa
3. Rysy charakterizujúce syntakticky závislé členy (subjekt, objekt a adverbiály)

V nasledujúcom texte všetky tieto skupiny rysov podrobne popíšeme. Zároveň vyslovíme presné návody, ako jednotlivé hodnoty rysov získať z výstupov morfologickej analýzy (v podobe morfologických značiek korpusu Penn Tree Bank [47]) a z výstupov syntaktickej analýzy (v podobe stanfordských závistlostí [41]).

6.1.1 Charakteristika cieľového slovesa

Rysy v tejto skupine reprezentujú dostupné informácie o cieľovom slovese. Skupina obsahuje celkovo 10 binárnych rysov, tj. rysov, ktoré nadobúdajú jednu z hodnôt $\{0, 1\}$. Vo väčšine prípadov daný rys vyjadruje svojou hodnotou informáciu, že vo vete existuje určitá syntaktická závislosť, resp. či dané cieľové sloveso má priradenú určitú morfologickú značku.

V tejto skupine rozoznávame týchto 10 rysov:

1. **Pasívnosť** (`passive_voice`). Rys informuje o existencii závislosti

   `auxpass`(TV, *).

   V zápise závislostí označujeme pojmom TV cieľové sloveso (*target verb*) a znakom * zasa ľubovoľný slovný token.

2. **Modalita 1** (`modality_1`). Rys informuje o existencii závislosti

$$\texttt{aux}(TV, \{would,\ should\}).$$

V zápise závislostí označujeme pojmom {*would*, *shloud*} možné slovné tokeny, ktoré pripúšťame na mieste závislého slovného tokenu.

3. **Modalita 2** (`modality_2`). Rys informuje o existencii závislosti

$$\texttt{aux}(TV, \{can,\ could,\ may,\ must,\ ought,\ might\}).$$

4. **Negácia** (`negation`). Rys informuje o existencii závislosti

$$\texttt{neg}(TV, *).$$

5. **Príčastie minulé** (`tense_vbn`). Rys informuje o tom, či cieľové sloveso je ohodnotené morfologickou značkou `VBN`. Touto značkou sú ohodnocované slovesá v tvare príčastia minulého.

6. **Minulý čas** (`tense_vbd`). Rys informuje o tom, či cieľové sloveso je ohodnotené morfologickou značkou `VBD`. Touto značkou sú ohodnocované slovesá v minulom čase.

7. **Príčastie prítomné** (`tense_vbg`). Rys informuje o tom, či cieľové sloveso je ohodnotené morfologickou značkou `VBG`. Touto značkou sú ohodnocované slovesá v tvare príčastia prítomného.

8. **Prítomný čas okrem tretej osoby** (`tense_vbp`). Rys informuje o tom, či cieľové sloveso je ohodnotené morfologickou značkou `VBP`. Touto značkou sú ohodnocované slovesá vo všetkých osobách okrem tretej, v prítomnom čase.

9. **Základná forma slovesa** (`tense_vb`). Rys informuje o tom, či cieľové sloveso je ohodnotené morfologickou značkou `VB`. Touto značkou sú ohodnotené slovesá v základnom tvare.

10. **Fráza v neurčitku** (`infinitive`). Rys informuje o existencii závislosti

$$\texttt{xcomp}(*,\ TV).$$

6.1.2 Charakteristika najbližšieho kontextu

Ako najbližší kontext cieľového slovesa sú definované 3 slová pred cieľovým slovesom a 3 slová za cieľovým slovesom. Pre každé takéto slovo je definovaných 9 binárnych rysov, tj. rysov, ktoré nadobúdajú jednu z hodnôt $\{0, 1\}$. Technické názvy rysov uvádzané v zátvorkách sú spoločné pre všetkých 6 kontextových slov. Kvôli jednoznačnej identifikácii sú navyše doplnené o prefixy `3p_`, `2p_`, `1p_`, `1n_`, `2n_` a `3n_` v poradí z ľava do prava.

1. **Podstatné mená** (`nominal`). Rys informuje o tom, či dané kontextové slovo je ohodnotené jednou z nasledujúcich morfologických značiek: `NN`, `NNS`, `NNP`, `NNPS`, `DT`, `PDT`, `PRP`, `PRP$`, `POS`, `CD`.

2. **Prídavné mená** (`adjective`). Rys informuje o tom, či dané kontextové slovo je ohodnotené jednou z nasledujúcich morfologických značiek: `JJ`, `JJR`, `JJS`.

3. **Slovesá** (`verbs`). Rys informuje o tom, či dané kontextové slovo je ohodnotené jednou z nasledujúcich morfologických značiek: `VB`, `VBD`, `VBG`, `VBN`, `VBP`, `VBZ`.

4. **Modálne slovesá** (`modal`). Rys informuje o tom, či dané kontextové slovo je ohodnotené morfologickou značkou `MD`.

5. **Príslovky** (`adverbial`). Rys informuje o tom, či dané kontextové slovo je ohodnotené jednou z nasledujúcich morfologických značiek: `RB`, `RBR`, `RBS`, `RP`, `IN`.

6. **Slovo *to*** (`to`). Rys informuje o tom, či dané kontextové slovo je ohodnotené morfologickou značkou `TO`.

7. **Opytovacie zámená** (`wh-pronoun`). Rys informuje o tom, či dané kontextové slovo je ohodnotené jednou z nasledujúcich morfologických značiek: `WDT`, `WP`, `WP$`.

8. **wh-adverb** (`wh-adverb`). Rys informuje o tom, či dané kontextové slovo je ohodnotené morfologickou značkou `WRB`.

9. **Slovo *be*** (`be`). Rys informuje o tom, či dané kontextové slovo má v lemmatizovanom tvare podobu *be*, tj. či sa jedná o nejaký tvar slovesa *to be*.

6.1.3 Charakteristika syntakticky závislých členov

V tejto skupine je definovaných niekoľko podskupín rysov, ktoré sa týkajú vždy určitého vetného člena, ktorý je priamo podradený cieľovému slovesu.

Subjekt

V tejto skupine sú zahrnuté informácie o subjekte. Obsahuje 3 binárne rysy.

1. **Nominálny podmet** (`n_subj`). Rys informuje o existencii závislosti

   $$\texttt{nsubj}(TV, *).$$

   Okrem toho môže rys nadobudnúť kladnú hodnotu aj v prípade existencie jednej zo závislostí

   $$\texttt{xsubj}(TV, *) \text{ alebo } \texttt{agent}(TV, *)$$

   pričom navyše podradený vetný člen musí byť ohodnotený jednou z nasledujúcich morfologických značiek: `NN`, `NNS`, `NNP`, `NNPS`, `CD`, `WDT`, `WP`.

2. **Klauzálny podmet** (`c_subj`). Rys informuje o existencii závislosti

   $$\texttt{csubj}(TV, *).$$

   Okrem toho môže rys nadobudnúť kladnú hodnotu aj v prípade existencie jednej zo závislostí

   $$\texttt{xsubj}(TV, *) \text{ alebo } \texttt{agent}(TV, *)$$

   pričom navyše podradený vetný člen nesmie byť ohodnotený ani jednou z nasledujúcich morfologických značiek: `NN`, `NNS`, `NNP`, `NNPS`, `CD`, `WDT`, `WP`.

3. **Podmet v množnom čísle** (`subj_pl`). Rys informuje o tom, či vo vete existuje subjekt, a ak áno, či je v množnom čísle. Technicky musí byť daný vetný člen ohodnotený morfologickou značkou `NNS` alebo `NNPS`.

Objekt

Ďalšia skupina ôsmych binárnych rysov reprezentuje informácie o objekte vo vete.

1. **Priamy predmet** (`dobj`). Rys informuje o existencii závislosti

$$\texttt{dobj}(TV, *).$$

2. **Nepriamy predmet** (`iobj`). Rys informuje o existencii závislosti

$$\texttt{iobj}(TV, *).$$

3. **Nominálny podmet v pasíve** (`nsubjpass`). Rys informuje o existencii závislosti

$$\texttt{nsubjpass}(TV, *).$$

4. **Klauzálny podmet v pasíve** (`csubjpass`). Rys informuje o existencii závislosti

$$\texttt{csubjpass}(TV, *).$$

5. **Klauzálny doplnok** (`ccomp`). Rys informuje o existencii závislosti

$$\texttt{ccomp}(TV, *).$$

6. **Komplement** (`complm`). Rys informuje o existencii závislosti

$$\texttt{complm}(TV, *).$$

7. **Objekt** (`any_obj`). Rys informuje o existencii objektu vo vete. Technicky rys nadobúda kladnú hodnotu v prípade, že existuje aspoň jedna zo závislostí popísaných v bodoch 1 až 6.

8. **Objekt v množnom čísle** (`obj_pl`). Rys informuje o tom, či vo vete existuje objekt, a ak áno, či je v množnom čísle. Technicky musí byť daný vetný člen ohodnotený morfologickou značkou `NNS` alebo `NNPS`.

Predložky frázových slovies

V tejto skupine je definovaný len jeden rys, ktorý ako prvý z množiny morfologicko-syntaktických rysov nie je binárny, ale kategoriálny. Znamená to, že môže nadobúdať jednu z dopredu nadefinovaných hodnôt. Hodnoty, v tomto prípade, predložky frázových slovies, sa získajú ešte pred utvorením vektorov rysov z trénovacích údajov.

Technicky tento rys označujeme ako `prt`. Označenie je odvodené od označenia stanfordskej závislosti, v ktorej je cieľové sloveso nadradeným tokenom a predložka je podradeným tokenom:

$$\texttt{prt}(TV, *).$$

Okrem kategoriálnych hodnôt, ktoré budú reprezentovať jednotlivé predložky, definujeme ešte 2 ďalšie možné hodnoty, ktoré môže rys nadobúdať. Hodnotu `none` nadobudne v prípade, že v inštancii neexistuje popisovaná závislosť, tj. cieľové sloveso nie je frázové a hodnotu `other` nadobudne v prípade, že sa v inštancii vyskytne závilosť s predložkou, ktorú nemáme v množine možných hodnôt tohto kategoriálneho rysu.

Adverbiály

V tejto skupine sú definované 4 binárne a 3 kategoriálne rysy, ktoré informujú o prítomnosti adverbiálov v danej inštancii.

1. **Príslovka 1** (`advmod`). Rys informuje o existencii závislosti

   $$\texttt{advmod}(TV, *).$$

2. **Príslovka 2** (`advcl`). Rys informuje o existencii závislosti

   $$\texttt{advcl}(TV, *).$$

3. **Príslovka 3** (`purpcl`). Rys informuje o existencii závislosti

   $$\texttt{purpcl}(TV, *).$$

4. **Príslovka 4** (`tmod`). Rys informuje o existencii závislosti

   $$\texttt{tmod}(TV, *).$$

5. **Adverbiálna predložka 1** (`prep`). Kategoriálny rys môže nadobúdať jednu z hodnôt, ktoré reprezentujú predložky extrahované zo závislostí

   $$\texttt{prep\_*}(TV, \dagger).$$

   V zápise závislosti znak * reprezentuje konkrétnu predložku a znak † ľubovoľný token. Pre každú predložku je definovaná samostatná závislosť (napríklad `prep_to` alebo `prep_with`). Podrobnosti môže čitateľ nájsť v dodatku B tejto práce. Okrem hodnôt reprezentujúcich konkrétne predložky môže tento rys nadobúdať ešte hodnotu `none`, ak v inštancii neexistuje ani jedna z popísaných závislostí a hodnotu `other` ak sa v inštancii vyskytne predložka, ktorá nie je obsiahnutá v množine možných hodnôt tohto kategoriálneho rysu. Konkrétne hodnoty, v tomto prípade, predložky, sa získajú ešte pred utvorením vektorov rysov z trénovacích údajov.

6. **Adverbiálna predložka 2** (`prepc`). Kategoriálny rys môže nadobúdať jednu z hodnôt, ktoré reprezentujú predložky extrahované zo závislostí

   $$\texttt{prepc\_*}(TV, \dagger).$$

   Všetky zásady pre tvorbu tohto rysu sú analogické s rysom Adverbiálna predložka 1.

7. **Marker** (`mark`). Kategoriálny rys môže nadobúdať jednu z hodnôt, ktoré reprezentujú predložky extrahované zo závislostí

   $$\texttt{mark}(TV, *).$$

   V zápise závislosti znak * reprezentuje konkrétnu predložku. Konkrétne predložky, ktoré budú tvoriť množinu možných hodnôt, ktoré môže rys nadobúdať, získame ešte pred vytváraním vektorov rysov z trénovacích údajov. Okrem hodnôt reprezentujúcich predložky môže tento rys, podobne, ako predošlé kategoriálne rysy, nadobúdať ešte hodnotu `none`, ak v inštancii neexistuje ani jedna z popísaných závislostí a hodnotu `other`, ak sa v inštancii vyskytne predložka, ktorá nie je obsiahnutá v množine možných hodnôt tohto kategoriálneho rysu.

6.2 Sémantické rysy

Hlavnou súčasťou definície jednotlivých patternov sú sémantické typy. Reprezentujú informáciu, aké podstatné mená sa môžu vyskytovať na určitých syntaktických pozíciách vo vete s daným cieľovým slovesom. Základnou myšlienkou vytvorenia sémantických rysov je snaha reprezentovať túto sémantickú informáciu vhodným spôsobom prostredníctvom vektorov rysov.

6.2.1 Možné prístupy vytvárania sémantických rysov

V ideálnom prípade by sme mali k dizpozícií klasifikátor sémantických typov, ktorý by každé podstatné meno klasifikoval do jedného, prípadne viacerých sémantických typov slovníka PDEV.

Rozpoznávač menných entít (NER)

Pred návrhom sémantických rysov sme sa rozhodli využiť výstup NER [46], ktorý o každom tokene v konkordancii rozhodne, či sa jedná o vlastné meno osoby, názov organizácie alebo o geografickú lokáciu.

Definovali sme 6 binárnych rysov – 3 pre subjekt a 3 pre objekt. Každý z týchto rysov reprezentuje odpoveď na otázku, či je subjekt alebo objekt inštancie ohodnotený niektorou z výstupných značiek rozpoznávača. Podrobnosti o tom, ako sme v inštancii identifikovali subjekty a objekty, uvádzame v ďalšej časti tejto kapitoly, venovanej sémantickým rysom.

Rozpoznávame nasledujúce 3 binárne rysy pre subjekt a objekt:

1. **Vlastné meno osoby** (`subj_person`, `obj_person`). Rys nadobúda kladnú hodnotu, ak je subjekt inštancie ohodnotený značkou `PERSON` vo výstupe rozpoznávača menných entít.

2. **Názov organizácie** (`subj_organization`, `obj_organization`). Rys nadobúda kladnú hodnotu, ak je subjekt inštancie ohodnotený značkou `ORGANIZATION` vo výstupe rozpoznávača menných entít.

3. **Geografické miesto** (`subj_location`, `obj_location`). Rys nadobúda kladnú hodnotu, ak je subjekt inštancie ohodnotený značkou `LOCATION` vo výstupe rozpoznávača menných entít.

Po navrhnutí týchto rysov sme zisťovali, koľko konkordancií má na pozícii subjektu alebo objektu slovo rozpoznané aplikáciou NER. Zistili sme, že na pozícii objektu sa vlastné podstatné mená nevyskytujú ani v jednej konkordancii v celej kolekcii VPS. Toto zistenie nám umožnilo automaticky vynechať tri nové rysy pre objekt, pretože by pre každú konkondanciu obsahovali nulové hodnoty.

V ďalších experimentoch sme preto využili len tri značky pre subjekt. Množinu týchto troch rysov sme označili ako **NER**.

Využitie WordNetu

Inou možnosťou, ktorú sme v priebehu experimentov testovali, bolo použiť nejaký existujúci sémantický zdroj, ktorý obsahuje veľkú databázu podstatných mien, ktoré sú navyše sémanticky anotované. Ako ideálny sa nám zdal byť WordNet [11]. Databáza (nie len) podstatných mien, nad ktorými je navyše vytvorená hierarchická štruktúra v podobe hyperonomických konceptov (teda slov všeobecnejších). Tieto hyperonomické koncepty sa vo WordNete nazývajú synsety. Hlavnou prekážkou použitia sa stala práve

priveľká množina synsetov. Chýbalo mapovanie, ktoré by synsety WordNetu namapovalo na sémantické typy v PDEV.

V našich experimentoch sme sa pokúšali vytvoriť jednak priame mapovanie synsetov na sémantické typy a jednak použiť nejakú existujúcu ontológiu, ktorá vytvára nejaký vhodný a malý vrchol hyperonomickej hierarchie. Takáto onológia bola vytvorená pre projekt EuroWordNet a publikovaná v [49]. Hlavnou prekážkou jej použitia bolo, že sa nám nepodarilo zistiť presné definície konceptov a pretovšekým mapovanie, ktoré boli použité pre zlúčenie jednotlivých synsetov do nových sémantických kategórií.

Populácie sémantických typov

Ďalšou možnosťou by mohlo byť vytvorenie zoznamu slov, ktoré patria k určitému sémantickému typu. Takýto experiment už bol implementovaný v práci [14]. P. Hanks hovorí o týchto zoznamoch ako o tzv. *populácii* sémantických typov. Vytvorené zoznamy by však museli byť veľmi obsiahle, aby dokázali pokryť všetky podstatné mená vyskytujúce sa v korpuse BNC50. Neexistuje ani jednoznačný algoritmus, ktorým by sa tieto populácie dali vytvoriť. Viac o tomto probléme píšeme v časti 4.3.

6.2.2 Lexikón sémantických prototypov (LSP)

Ako zdroj sémantickej informácie pre klasifikátory patternov sme nakoniec vybrali Lexikón sémantických protypov (LSP). Originálna myšlienka LSP bola prvý krát publikovaná v [50]. Metodológia LSP bola odvtedy použitá pre viacero jazykov vrátane angličtiny.

LSP je databáza podstatných mien, v ktorej je každé podstatné meno anotované minimálne jedným *sémantickým prototypom*. Zoznam sémantických prototypov uvádzame v dodatku C a myslíme si, že sú natoľko podobné sémantickým typom v slovníku PDEV, že by mohli byť dobrou aproximáciou sémantickej informácie pre naše klasifikátory.

Sémantické prototypy v LSP tvoria množinu približne 200 sémantických značiek. Sú zatriedené do niekoľkých skupín. V každej skupine je definovaný všeobecnejší prototyp, ktorý označujeme ako **zastrešujúci prototyp** (*umbrella prototype*) a ktorým možno aproximovať významy všetkých podstatných mien anotovaných niektorým z prototypov z danej skupine. Orientáciu medzi zastrešujúcimi prototypmi a *bežnými* prototypmi v danej skupine uľahčuje konvencia pri pomenovávaní protypov. Všetky prototypy v jednej skupine majú rovnaký prefix, ktorý je zároveň označením zastrešujúceho prototypu danej skupiny. Napríklad zastrešujúci prototyp pre podstatné mená zo zoológie má označenie <**A**>. Ďalšie prototypy v skupine zoológia sú potom označené ako <**Adom**>, <**Aich**>, <**Amyth**>, a pod.

Navyše je v LSP definovaná možnosť zjednotenia zastrešujúcich prototypov do ešte všeobecnejších 22 skupín, ktoré reprezentujú 22 základných konceptov. Jedná sa o najvšeobecnejšie protypy. Bick ich označuje ako **hlavné zastrešujúce prototypy** (*major umbrellas*). Ich definície môže čitateľ nájsť v dodatku C.

LSP obsahuje okrem všeobecných podstatných mien aj bohatú databanku vlastných podstatných mien, ktoré automaticky anotoval rozpoznávač menných entít (NER, z anglického *Named-entity recognizer*). Vlastné podstatné mená majú definovanú vlastnú sadu sémantických značiek (kvôli kompatibilite s ďalšími projektami). Niektoré z týchto značiek je možné automaticky zaradiť do definovaných zastrešujúcich prototypov vďaka spoločnému prefixu a istá množina značiek je už v dokumentácii zaradená do systému prototypov.

V LSP rozlišujeme tri základné časti:

1. Bezpečný segmet (SS)

2. Vlastné podstatné mená (NER)

3. Ďalšie všeobecné podstatné mená

Približne 14000 najčastejších slov označujeme ako bezpečný segment (SS, z anglického *safe segment*). Keďže lexikón je stále vo vývoji, autori LSP sa rozhodoli vytvoriť SS, ktorý obsahuje najčastejšie podstatné mená a ktoré boli ručne zrevidované najskôr.

Druhý segment lexikónu predstavuje lexikón vlastných podstatných mien (NER). Obsahuje približne 22000 slov, ktoré sú anotované špeciálnymi značkami používanými rozpoznávačom menných entít. Ich zoznam a presné definície nájde čitateľ v dodatku C.

Zvyšná a najväčšia časť LSP zatiaľ nebola ručne prečistená a môžu sa v nej vyskytovať nekonzistencie a niektoré nedefinované sémenatické prototypy, ktoré boli v minulosti zavedené a neskôr opäť zrušené. Predstavuje najväčšiu časť lexikónu s približne 36000 slovami.

Základné štatistiky LSP

	SS		**SS + NER**		**ALL**	
	Dokumentované	**Všetky**	**Dokumentované**	**Všetky**	**Dokumentované**	**Všetky**
Slov	13455	14216	38676	39465	73802	75796
Prototypov	179	295	183	300	186	345
Prototypov na slovo	1.3	1.4	1.6	1.6	1.4	1.4
Najmenšia populácia	1	1	1	1	1	1
Najväčšia populácia	1571	1571	23379	23379	23381	23381
Priemerná popoulácia	100.0	66.5	336.1	210.9	538.2	300.0

Tabuľka 6.1: Základné štatistiky LSP

Tabuľka 6.1 uvádza základné štatistiky LSP. Je rozdelená do troch sekcií. V prvej sekcii berieme do úvahy len bezpečný segment (SS), v druhej sekcii sme k bezpečnému segmentu pridali vlastné podstatné mená (SS+NER) a v poslednej časti tabuľky uvádzame štatistiky pre všetky lexikálne jednotky v lexikóne (ALL).

Pre každú sekciu lexikónu sme navyše vytvorili dve štatistické charakteristiky. Prvá s označením *Dokumentované* započítava len prototypy, ktoré sú definované v dokumentácii sémantických typov v [51]. Ak slovo nemalo priradený žiaden oficiálne definovaný prototyp, vyradili sme ho z lexikónu a ďalej s ním nepracovali. Druhá charakteristika s označením *Všetky* započítava všetky slová a všetky prototypy v lexikóne, i tie, ktoré neboli definované v dokumentácii.

Zaujímalo nás tiež, aké množstvo slov je anotovaných viac ako jedným sémantickým prototypom, tj. *ambiguita* slov v lexikóne. Tento aspekt lexikónu je uvedený v tabuľke 6.2, v ktorej je pre každý uvažovaný segment uvedený frekvenčný zoznam slov podľa počtu protytov, ktorými sú slová anotované. Z tabuľky vyplýva, že väčšina slov má priradený len jeden prototyp, ale nezanedbateľné množstvo slov je anotované ambiguitne dvoma alebo viacerými prototypmi.

	SS		SS + NER		ALL	
Počet prototypov	**Dokumentované**	**Všetky**	**Dokumentované**	**Všetky**	**Dokumentované**	**Všetky**
1	9718	9775	19762	20251	49537	50477
2	3120	3625	17475	17558	22512	23266
3	528	696	1280	1453	1523	1765
4	85	110	139	174	194	241
5	4	10	15	24	30	41
6	0	0	5	5	5	5
10	0	0	0	0	1	1

Tabuľka 6.2: Ambiguita prototypov LSP

Najdôležitejšou úlohou pred použitím LSP bolo zistiť, ako dobre tento sémantický zdroj pokrýva subjekty a objekty konkordancií v kolekcii VPS. Pre úplnosť dodávame, že počas celého experimentu sme pracovali len s údajmi určenými na trénovanie klasifikátorov, údaje určené na testovanie sme v čase experimentu stále nemali k dizpozícii.

Pred samotným vyhľadávaním slov v lexikóne sme najskôr museli identifikovať subjekty a objekty v konkordanciách podobne, ako sme to robili pri identifikácii morfosyntaktických rysov `n_subj` a `any_obj` v časti 6.1. Z takto identifikovaných tokenov sme najskôr vylúčili osobné zámená, ktoré lexikón neobsahuje, ale ktoré môžme s istotou zaradiť automaticky do prototypu <**H**> (reprezentujúceho ľudské bytosti).

V prípade subjektu sme ešte navyše medzi subjekty typu <**H**> započítali aj inštancie, v ktorých sme síce neidentifikovali subjekt, ale ktoré boli v pasíve. Učinili sme tak na základe pozorovania skúseného anotátora konkordancií VPS, ktorý si všimol, že inštancie v pasíve, ktoré nemajú vyjadrené subjekty priamo, sa dajú veľmi často zaradiť do sémantického typu [[**Human**]] alebo [[**Institution**]].

Po odstránení týchto výnimiek nám ostali menné subjekty a objekty, ktoré sme sa snažili vyhľadávať v LSP. Výsledky predkladáme v tabuľke 6.3 a v dodatkoch D.4 a D.5.

V tabuľke 6.3 uvádzame pokrytie pre rôzne konfigurácie lexikónu – subjekty a objekty sme buď vyhľadávali len v určitých segmentoch lexikónu (SS, SS+NER, ALL) alebo sme používali rôzne množiny sémantických protypov (buď len zdokumentované, všetky, alebo len hlavné zastrešujúce prototypy). V prípade, že sme niektoré prototypy z množiny vypustili (typicky nezdokumentované), prestali sme brať do úvahy i slová, ktoré boli týmito prototypmi anotované, preto sa výsledky pokrytia mohli líšiť podľa toho, s ktorou množinou sémantických prototypov sme pracovali.

Riadok *Počet konkordancií* zobrazuje počet trénovacích konkordancií v kolekcii VPS. V každej z týchto konkordancií sme vyhľadávali subjekt a objekt a, ak existoval, pokúsili sme sa ho vyhľadať v lexikóne. Riadky *Počet subjektov* a *Počet objektov* zobrazujú počet konkordancií, v ktorých sme identifikovali subjekty a objekty. Na mieste subjektov, aj objektov sa mohli vyskytovať aj osobné zámená, ktoré sa v lexikóne nenachádzajú, uvádzame ich preto osobitne v riadkoch *Počet os. zámen.* Ako sme už popísali vyššie, v prípade subjektov sme započítavali aj konkordancie v pasíve, ktoré nemali identifikovaný subjekt. Ich počet uvádzame v riadku *Počet pas. konkordancií.*

Segmenty	**SS**	**SS+NER**	**ALL**	**ALL**	**ALL**
Zdokumentované	**Áno**	**Áno**	**Áno**	**Nie**	**Nie**
Umbrellas	**Nie**	**Nie**	**Nie**	**Nie**	**Áno**
Počet konkordancií	10150	10150	10150	10150	10150
Počet subjektov	6386	6386	6386	6386	6386
Počet os. zámen	1192	1192	1192	1192	1192
Počet pas. konkordancií	1593	1593	1593	1593	1593
Počet nominálnych	3601	3601	3601	3601	3601
Nájdených v BSL	2278	2611	2732	2875	2860
Konkordancií so subjektom	62.92%	62.92%	62.92%	62.92%	62.92%
Zasiahnutých konkordancií	49.88%	53.16%	54.35%	55.76%	55.62%
Pokrytie lexikónu	63.26%	72.51%	75.87%	79.84%	79.42%
Počet objektov	6092	6092	6092	6092	6092
Počet os. zámen	297	297	297	297	297
Počet nominálnych	5795	5795	5795	5795	5795
Nájdených v BSL	3597	3679	3937	4155	4079
Konkordancií s objektom	60.02%	60.02%	60.02%	60.02%	60.02%
Zasiahnutých konkordancií	38.36%	39.17%	41.71%	43.86%	43.11%
Pokrytie lexikónu	62.07%	63.49%	67.94%	71.70%	70.39%

Tabuľka 6.3: Pokrytie kolekcie VPS vybranými segmentami LSP.

Najdôležitejšie riadky predstavujú počet subjektov a objektov, ktoré sme našli v lexikóne – *Nájdených v BSL*.

Riadky *Konkordancií so subjektom*, resp. *Konkordancií s objektom* udávajú podiel konkordancií, v ktorých sme identifikovali tieto vetné členy. V oboch prípadoch je to približne 60%. Riadky *Zasiahnutých konkordancií* zobrazujú podiel inštancí (z celkového počtu všetkých trénovacích konkordancií), ktorým budeme schopní vďaka sémantickým rysom upraviť vektory rysov o nové informácie. Zaujímavý je tiež riadok *Pokrytie lexikónu*, ktorý udáva koľko percent menných subjektov a objektov sa podarilo vyhľadať v lexikóne a priradiť im tak nejakú sémantickú značku.

Z tabuľky môžme vidieť, že najlepšie pokrytie sme dosiahli, ak sme subjekty a objekty vyhľadávali v celom lexikóne a s použitím všetkých (i nezdokumentovaných) sémantických prototypov. Na záver nás zaujímalo, ako sa zmení pokrytie, ak namiesto všetkých sémantických prototypov použijeme len zastrešujúce prototypy. Pokrytie mohlo klesnúť práve kvôli nezdokumentovaným prototypom, pre ktoré nie je definovaný žiaden zastrešujúci prototyp.

Výsledky experimentu môže čitateľ vidieť v poslednom stĺpci tabuľky 6.3. Po použití hlavných zastrešujúcich prototypov sme stratili len 0,42% z pokrytia lexikónu pre subjekty a 1,31% z pokrytia pre objekty. V absolútnych číslach to znamená, že použitím zastrešujúcich prototypov by sme stratili len 15 subjektov a 76 objektov. Myslíme si, že takáto strata je prijateľná vzhľadom k výraznému zníženiu počtu sémantických značiek, ak budeme pracovať len s 22 hlavnými zastrešujúcimi prototypmi.

Podrobné výsledky experimentu s pokrytím subjektov pre všetky slovesá uvádzame v dodatku D.4. Tabuľka vyjadruje pokrytie kolecie VPS-30-En lexikónom pracujúcim so všetkými slovami v lexikóne a so všetkými (nie len zdokumentovanými) prototypmi, ktoré sme následne zlúčili až na úroveň hlavných zastrešujúcich prototypov.

Podrobné výsledky experimentu s pokrytím objektov pre všetky slovesá uvádzame v dodatku D.5. Tabuľka vyjadruje pokrytie kolekcie VPS lexikónom pracujúcim so všetkými slovami v lexikóne a so všetkými (nie len zdokumentovanými) prototypmi,

ktoré sme následne zlúčili až na úroveň hlavných zastrešujúcich prototypov.

Dve sady sémantických rysov

Na základe uvedených experimentov sme sa rozhodli vytvoriť sadu 22 sémantických rysov, ktoré zodpovedajú 22 hlavným zastrešujúcim prototypom. Tabuľka 6.4 predstavuje hlavné zastrešujúce prototypy. Celkovo definujeme 44 sémantických rysov – 22 pre subjekty (s prefixom `subj`) a 22 pre objekty (s prefixom `obj`). Túto množinu sémantických rysov budeme označovať ako **MU44** (*Major Umbrellas 44*).

Rys	**Rys**
Zoológia (`A`)	Celky a kolektívy (`coll`)
Botanika (`B`)	Domény (`domain`)
Ľudské bytosti (`H`)	Vlastnosti (`f`)
Lokácie (`L`)	Jedlo (`food`)
Dopravné prostriedky (`V`)	Pocity a vnímanie (`percep`)
Abstraktné koncepty (`ac`)	Produkty ľudskej kreativity (`sem`)
Akcieudalostiprocesy (`act`)	Okolnosti dejov (`sit`)
Anatómia (`an`)	Čas (`temp`)
Vecikonkrétne koncepty (`cc`)	Nástroje a stroje (`tool`)
Oblečenie (`clo`)	Jednotky (`unit`)
Materiály (`cm`)	Počasie (`wea`)

Tabuľka 6.4: Prehľad 22 sémantických rysov, ktoré tvoria množinu **MU44**.

Okrem tejto sady sémantických rysov sme sa rozhodli experimentálne overiť aj ďalšiu – väčšiu sadu sémantických rysov. Pri jej definovaní sme vyšli z definície hlavných zastrešujúcich prototypov (sú definované v dodatku C). Tie sú definované ako zjednotenie vybraných zastrešujúcich prototypov. Novú sadu rysov sme vytvorili pomocou zastrešujúcich prototypov, ktoré sa vyskytovali v definíciách hlavných zastrešujúcich prototypov a ktoré sme sa rozhodli nezlučovať, ale nechať oddelené, ako samostatné rysy. Získali sme tak množinu 62 rysov pre subjekt a 62 rysov pre objekt.

Cieľom experimentu s väčším množstvom rysov je overiť, či zlúčením prototypov do 22 hlavných skupín nestrácame príliš veľa zo sémantickej informácie uloženej vo väčšej granularite prototypov.

Prehľad a popis sémantickych rysov tejto sady uvádzame v tabuľke 6.5. Množinu rysov sme nazvali **AU124** (*All Umbrellas 124*).

Rys	**Rys**
Zoológia (A)	Časti (part)
Botanika (B)	Kúsky (piece)
Ľudské bytosti (H)	Domény (domain)
Organizácie (org)	Ideológie (ism)
Inštitúcie (inst)	Žánre (genre)
Mená osôb (hum)	Jazyky (ling)
Ženské mená (fem)	Športy (sport)
Mužské mená (masc)	Hry (game)
Historické mená (Hfunc)	Vlastnosti (f)
Lokácie (L)	Farby (col)
Budovy (build)	Jedlo (food)
Smery (dir)	Pitie (drink)
Dopravné prostriedky (V)	Obovie (fruit)
Abstraktné prostriedky (ac)	Vnímanie (percep)
Abstraktné nepočítateľné slová (am)	Produkty ľudskej kreativity (sem)
Geometrické tvary (geom)	Pravidlázákony (conv)
Meta substantíva (meta)	Situácie (sit)
Akcie (act)	Stavy (state)
Udalosti (event)	Polohy (pos)
Procesy (process)	Systémy (system)
Príležitosti (occ)	Čas (temp)
Diskusie (talk)	Periodické udalosti (per)
Konflikty (fight)	Mesiace (month)
Anatómia (an)	Nástroje (tool)
Konkrétne počítateľné veci (cc)	Stroje (mach)
Kontajnery (con)	Jednotky (unit)
Nábytok (furn)	Meny (cur)
Oblečenie (clo)	Trvanie (dur)
Konkrétne nepočítateľné substancie (cm)	Množstvá (amount)
Materiály (mat)	Peniaze (mon)
Kolektív, skupina (coll)	Počasie (wea)

Tabuľka 6.5: Prehľad 62 sémantických rysov, ktoré tvoria množinu **AU124**.

Kapitola 7

Príprava údajov pre strojové učenie

V tejto kapitole popisujeme experimenty, ktoré sme vykonali pred započatím experimentov so strojovým učením. Začíname popisom získavania možných hodnôt pre kategoriálne rysy, pokračujeme popisom rozdelenia dostupných inštancií na trénovacie a testovacie. Na záver kapitoly stanovujeme baseline pre ďalšie experimenty.

7.1 Hodnoty kategoriálnych rysov

V množine morfo-syntaktických rysov, ktorá je podrobne popísaná v sekcii 6.1 tejto práce, sú definované 4 kategoriálne rysy:

1. Predložka frázového slovesa (`prt`)
2. Adverbiálna predložka 1 (`prep`)
3. Adverbiálna predložka 2 (`prepc`)
4. Marker (`mark`)

Všetky možné hodnoty týchto kategoriálnych rysov sme sa rozhodli extrahovať z dostupných trénovacích inštancií. Nie všetky extrahované hodnoty ale má zmysel použiť ako možné hodnoty pre kategoriálne rysy. Ak sa niektorá z hodnôt vyskytuje s veľmi malou frekvenciou, algoritmy strojového učenia budú mať príliš málo trénovacích príkladov na to, aby z danej hodnoty získali nejakú informáciu potrebú pre klasifikáciu patternov.

Pre každý zo štyroch kategoriálnych rysov sme vytvorili frekvenčný zoznam možných hodnôt, ktoré sa vyskytli v trénovacích inštanciách. Na zaradenie do množiny možných hodnôt stačilo, aby daná hodnota bola dostatočne frekventovaná aspoň v jednom z 30 slovies v kolekcii. Sledovali sme preto, na akom slovese dosiahla hodnota maximálnu frekvenciu.

Po zvážení výsledkov experimentu sme sa rozhodli zvoliť ako prahovú frekvenciu hodnotu 10. Znamená to, že ak sa hodnota rysu vyskytovala v trénovacích inštanciách menej ako 10-krát, nezaradili sme ju do množiny možných hodnôt daného kategoriálneho rysu. Takéto málo frekventované hodnoty sa podľa definície nahradili hodnotou `other`, ktorú sme definovali ako hodnotu, ktorú môže nadobudnúť každý zo štyroch kategoriálnych rysov.

Poslednú úpravu, ktorú sme po získaní zoznamu hodnôt kategoriálnych rysov urobili, bolo prevedenie týchto rysov na binárne. Prevedenie bolo nutné kvôli geometrickým algoritmom strojového učenia, ktoré nedokážu pracovať s kategoriálnymi rysmi.

Prevod kategoriálneho rysu na sériu binárnych rysov predstavuje triviálnu úlohu. Pre každú hodnotu kategoriálneho rysu sa definuje jeden binárny rys, ktorý informuje o tom, či kategoriálny rys nadobudol danú hodnotu.

V ďalšom texte sa zaoberáme každým kategoriálnym rysom individuálne a na základe zostavených frekvenčných zoznamov definujeme množinu možných hodnôt pre daný rys.

Predložka frázového slovesa (`prt`)

Výsledky experimentu na meranie frekvencie jednotlivých predložiek prezentujeme v tabuľke 7.1.

Hodnota	Maximum	Sloveso	Hodnota	Maximum	Sloveso
up	252	yield	over	10	yield
out	180	yield	around	7	yield
down	58	yield	together	4	yield
away	57	trouble	open	3	trouble
in	49	yield	across	2	trouble
off	43	yield	about	1	trouble
back	38	yield	forth	1	pour
on	12	yield	through	1	trouble

Tabuľka 7.1: Frekvenčný zoznam predložiek frázových slovies vyskytujúcich sa v trénovacích inštanciách.

Z uvedenej tabuľky vyplýva, že najčastejšie sa predložky vyskytovali pri slovese *yield*. Množina možných hodnôt rysu Predložka frázového slovesa bude teda obsahovať tieto hodnoty:

{*none*, *other*, *up*, *out*, *down*, *away*, *in*, *off*, *back*, *on*, *over*}

Po prevedení rysu do binárnej podoby teda získame 11 binárnych rysov.

Adverbiálna predložka 1 (`prep`)

Výsledky experimentu na meranie frekvencie jednotlivých predložiek prezentujeme v tabuľke 7.2. V tabuľke z dôvodu úspory miesta prezentujeme len predložky, ktoré sa dostali nad stanovenú prahovú hranicu 10 výskytov.

Z uvedenej tabuľky vyplýva, že najčastejšie sa predložky vyskytovali pri slovese *yield*. Po prevedení rysu do binárnej podoby získame 36 binárnych rysov. Množina možných hodnôt rysu *Adverbiálna predložka 1* bude obsahovať tieto hodnoty:

{*none*, *other*, *in*, *to*, *with*, *into*, *on*, *at*, *from*, *for*, *as*, *by*, *of*, *through*, *after*, *over*, *out of*, *during*, *about*, *against*, *like*, *under*, *between*, *than*, *behind*, *within*, *without*, *around*, *away from*, *before*, *towards*, *upon*, *despite*, *across*, *beyond*, *out*}

Adverbiálna predložka 2 (`prepc`)

Výsledky experimentu na meranie frekvencie jednotlivých predložiek prezentujeme v tabuľke 7.3. V tabuľke z dôvodu úspory miesta prezentujeme len predložky, ktoré sa dostali nad stanovenú prahovú hranicu 10 výskytov.

Z uvedenej tabuľky vyplýva, že najčastejšie sa predložky vyskytovali pri slovese *yield*. Po prevedení rysu do binárnej podoby získame 11 binárnych rysov. Množina možných hodnôt rysu *Adverbiálna predložka 2* bude obsahovať hodnoty:

Hodnota	Maximum	Sloveso	Hodnota	Maximum	Sloveso
in	833	yield	against	43	yield
to	632	yield	like	40	yield
with	591	yield	under	39	yield
into	495	yield	between	34	yield
on	348	yield	than	22	yield
at	346	yield	behind	21	yield
from	271	yield	within	20	yield
for	257	yield	without	20	yield
as	228	yield	around	18	yield
by	170	yield	away_from	17	trouble
of	144	yield	before	17	yield
through	106	yield	towards	16	yield
after	57	yield	upon	15	yield
over	57	yield	despite	14	yield
out_of	54	yield	across	13	yield
during	50	yield	beyond	12	yield
about	46	yield	out	11	yield

Tabuľka 7.2: Frekvenčný zoznam adverbiálnych predložiek vyskytujúcich sa v trénovacích inštanciách

Hodnota	Maximum	Sloveso	Hodnota	Maximum	Sloveso
by	43	yield	after	16	yield
in	28	yield	into	16	submit
with	21	yield	without	15	yield
as	17	trouble	according_to	13	yield
for	17	yield			

Tabuľka 7.3: Frekvenčný zoznam klauzálnych adverbiálnych predložiek vyskytujúcich sa v trénovacích inštanciách

$$\{none, other, by, in, with, as, for, after, into, without, according\_to\}$$

Marker (`mark`)

Výsledky experimentu na meranie frekvencie jednotlivých markerov prezentujeme v tabuľke 7.4.

Z uvedenej tabuľky vyplýva, že najčastejšie markery sa vyskytovali pri slovese *yield*. Množina možných hodnôt rysu *Marker* bude teda obsahovať tieto hodnoty:

$$\{none, other, as, if, although, because, while, after, before, until, since, in\}$$

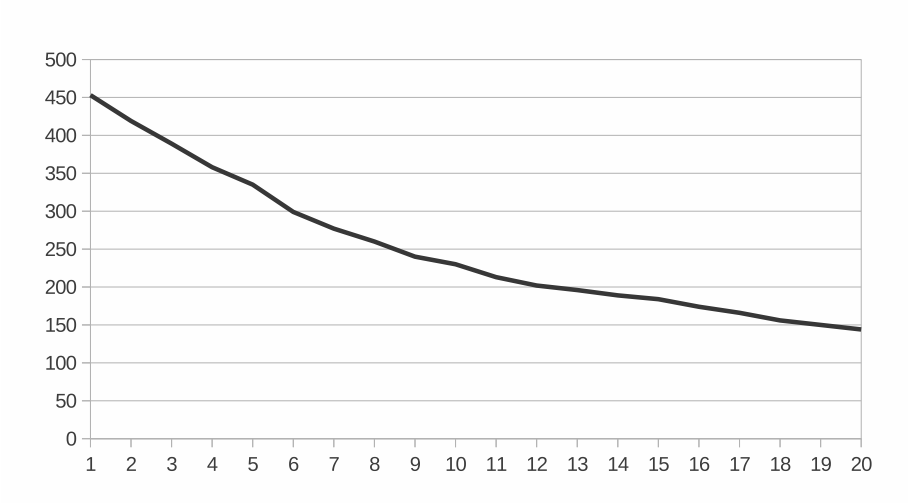
Po prevedení rysu do binárnej podoby získame 12 binárnych rysov.

7.2 Prahové frekvencie patternov

V tejto časti práce popisujeme experimenty, v ktorých sme sledovali rôzne charakteristiky inštancií a na základe ktorých sme sa rozhlodli stanoviť prahovú frekvenciu výskytu jednotlivých patternov, ktoré zahrnieme do procesu strojového učenia. Podrobnosti uvádzame v časti 5.2.

Hodnota	Maximum	Sloveso	Hodnota	Maximum	Sloveso
up	252	yield	over	10	yield
out	180	yield	around	7	yield
down	58	yield	together	4	yield
away	57	trouble	open	3	trouble
in	49	yield	across	2	trouble
off	43	yield	about	1	trouble
back	38	yield	forth	1	pour
on	12	yield	through	1	trouble

Tabuľka 7.4: Frekvenčný zoznam markerov vyskytujúcich sa v trénovacích inštanciách



Obr. 7.1: Zmena celkového počtu patternov pre rôzne hodnoty prahovej frekvencie.

V jednotlivých experimentoch sme sledovali nasledujúce charakteristiky slovies:

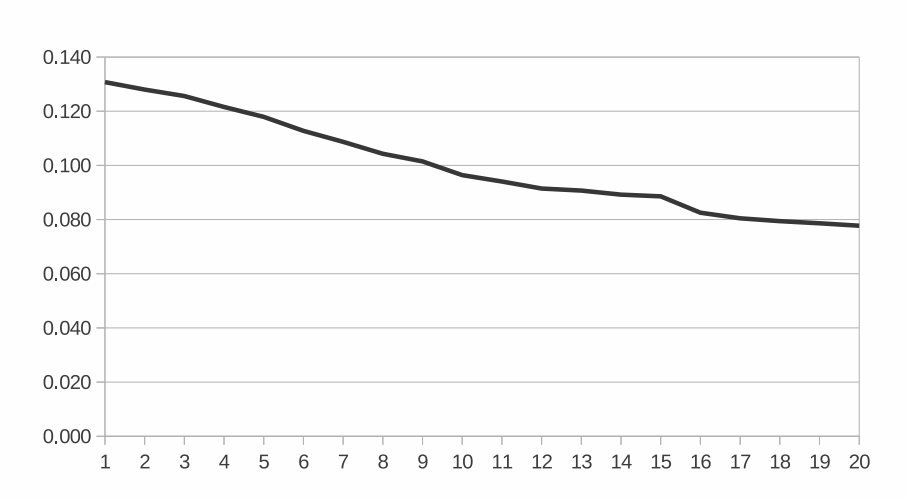
1. počet patternov, ktoré ostanú zachované po aplikácii prahovej frekvencie;
2. zmenu perplexity po aplikácii prahovej frekvencie;
3. počet inštancií, ktorým zmeníme pattern po aplikácii prahovej frekvencie; a
4. podiel inštancií, ktoré vložíme do patternu `u`.

Vo všetkých popisovaných experimentoch sme pracovali len s údajmi určenými na trénovanie.

Zmena počtu patternov po aplikácii prahovej frekvencie

V tomto experimente sme sledovali ako sa vyvíja počet patternov pre jednotlivé slovesá po aplikácii rôznych prahových frekvencií. Výsledky experimentu uvádzame v dodatku D.6, na strane 104. Pre úsporu miesta uvádzame počty patternov len pre niektoré hodnoty prahovej frekvencie.

V jednotlivých stĺpcoch tabuľky sú uvedené počty patternov, ktoré majú frekvenciu rovnú alebo väčšiu ako číslo uvedené v záhlaví tabuľky. Na posledných riadkoch tabuľky uvádzame, ako sa mení celkový počet patternov a aký je maximálny a minimálny počet patternov medzi jednotlivými slovesami.



Obr. 7.2: Zmena váženej perplexity pre celú kolekciu pre rôzne hodnoty prahovej frekvencie.

Na obrázku 7.1 uvádzame graf vývoja celkového počtu patternov pre jednotlivé hladiny prahovej frekvencie. Môžeme vidieť, že počet patternov klesá približne rovnomerne. Neexistuje teda žiadna zlomová hodnota, pri ktorej by sa počet patternov výrazne zmenil. Rozhodli sme sa preto skúmať ďalšie charakteristiky slovies.

Zmena perplexity pre rôzne prahové frekvencie

V tomto experimente nás zaujímalo, ako sa bude vyvíjať perplexita pre rôzne úrovne prahovej frekvencie. Výsledky uvádzame v dodatku D.7, na strane 105. Pre úsporu miesta uvádzame len niektoré hodnoty prahovej frekvencie.

Zo získaných perplexít pre jednotlivé slovesá sme váženým priemerom popísaným v časti 5.1.3 spočítali perplexitu celého súboru slovies. Vývoj takto spočítanej perplexity uvádzame na obrázku 7.2.

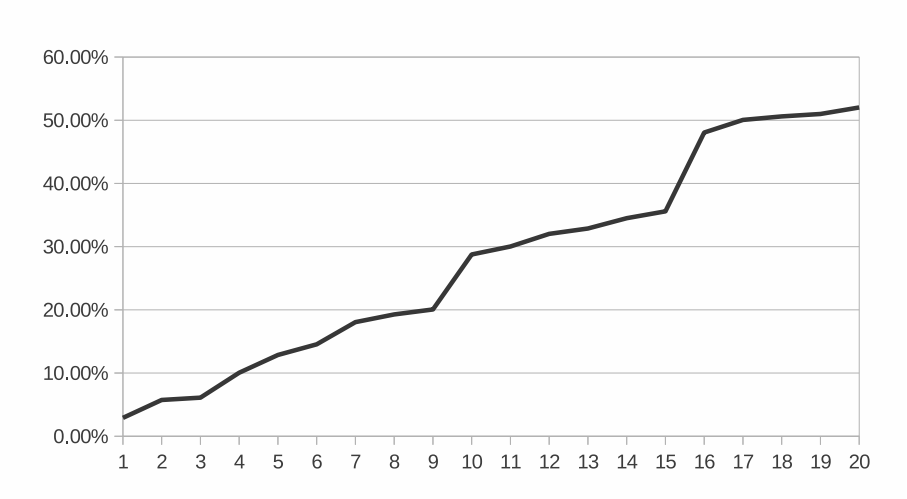
Z uvedeného vývoja perplexity je zrejmé, že pri žiadnej hodnote prahovej frekvencie nenastáva výrazná zmena v perplexite, podľa ktorej by sme prahovú frekvenciu mohli vhodne zvoliť. Rozhodli sme sa preto experimentovať s ďalšími charakteristikami kolekcie.

Počet inštancií so zmeneným patternom

V tomto experimente sme skúmali počet inštancií, ktoré budú nastavením prahovej frekvencie ovplyvnené, tj. zmeníme im pattern z nejakého čísla na značku `u`. Dodatok D.8, na strane 106 zobrazuje výsledky experimentu pre vybrané hodnoty prahovej frekvencie.

Ako prahovú frekvenciu sme zvolili hodnotu 9 po výsledkoch posledného experimentu, v ktorom sme sledovali, ako rastie priemerný počet inštancií anotovaných značkou `u` pre jednotlivé nastavenia prahovej frekvencie. Dodatok D.9 na strane 107 zobrazuje percentuálny podiel značky `u` vzhľadom k dostupnému počtu trénovacích inštancií pre každé sloveso.

Z percentuálnych podielov inštancií anotovaných značkou `u` sme spočítali vážený priemerný počet inštancií anotovaných značkou `u` cez všetky slovesá, tak ako ho popisujeme v časti 5.1.3 a priebeh tohto priemeru zobrazili na obrázku 7.3. Obrázok zobrazuje



Obr. 7.3: Zmena priemerného váženého počtu inštancií anotovaných značkou u v celej kolekcii pre rôzne prahové frekvencie.

zmenu priemerného váženého počtu inštancií anotovaných značkou u pre rôzne prahové frekvencie.

Z obrázku môžeme videť, že počet inštancií anotovaných značkou u postupne rastie približne rovnomerne. Medzi hodnotami prahovej frekvencie 9 a 10 ale skokovo vzrastá takmer o 10%. Na základe toho sme sa rozhodli, že ako prahovú frekvenciu zvolíme hodnotu 9. Značka u tak celkovo bude obsahovať 20% všetkých inštancií z trénovacích údajov.

Zaujímavou charakteristikou kolekcie VPS je tiež počet dostupných inštancií, ktoré sú priradené jednotlivým patternom. Na obrázku 7.4 sú jednolivé patterny (zo všetkých slovies), po aplikácii prahovej frekvencie, zoradené zostupne podľa počtu inštancií, ktoré boli danými patternami anotované. Môžeme zaznamenať výraznú riedkosť údajov. Len pre 50 patternov, z celkového počtu 240, máme k dispozícii aspoň 50 inštancií na trénovanie klasifikátorov. Považujeme to za veľmi negatívnu skutočnosť, ktorá sa môže výrazne podpísať pod celkovú úspešnosť modelov.

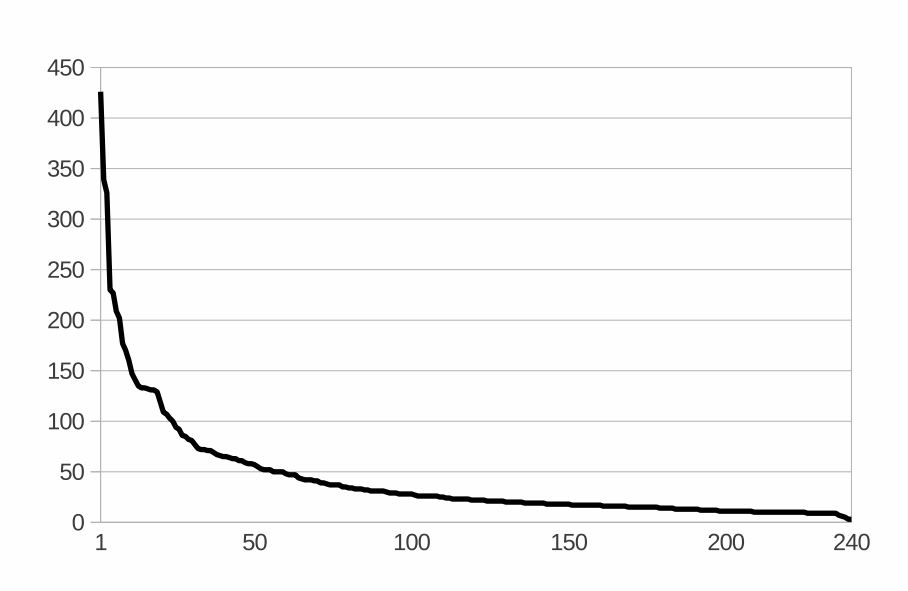
7.3 Rozdelenie inštancií na trénovacie a testovacie

Pri riešení úloh strojového učenia je nutné rozdeliť údaje na trénovacie a testovacie a dôsledne toto rozdelenie dodržovať počas celej doby experimentov. V trénovacej fáze môže prísť model do styku len s trénovacími údajmi. V opačnom prípade, tj. ak by model získal nejakú vedomosť aj z testovacích údajov, by sme nemali reprezentatívnu vzorku nevidených inštancií, na ktorých by sme mohli model otestovať.

V kolekcii VPS sme od začiatku mali jasnú predstavu, ktoré inštancie zvolíme za trénovacie a ktoré za testovacie. Kolekcia VPS obsahuje pre každé sloveso vzorku 50 multianotovaných inštancií, ktoré anotovali 4 anotátori a ktoré následne prešli adjudikačným procesom.

Inštancie, pre ktoré existuje len jedna anotácia a tvoria referenčnú vzorku sme sa rozhodli použiť ako trénovacie údaje. Počet trénovacích a testovacích údajov, ktoré máme k dispozícii je uvedený v tabuľke 5.1.

Vďaka takémuto rozdeleniu sme mohli porovnávať výsledky klasifikátora s ďalšími anotátormi a sledovať napríklad počet inštancií, na ktorých sa klasifikátor zhodol s aspoň jedným ľudským anotátorom.



Obr. 7.4: Počet konkordancií pre jednotlivé patterny.

7.4 Cross-validácia

Z tabuľky 5.1 vyplýva, že množina dostupných trénovacích inštancií je pre jednotlivé slovesá značne obmedzená. Štandardným postupom, ktorý sa v takejto situácii uplatňuje, je zavedenie cross-validácie.

Cross-validácia spočíva v tom, že namiesto trénovania, ladenia a testovania jedného modelu sa trénuje, ladí a testuje hneď niekoľko modelov. Každý z modelov je trénovaný na (o trochu) rozdielnych trénovacích inštanciách a zároveň každý z modelov je testovaný na disjunktných množinách testovacích údajov.

Pri n-fold cross-validácii sa všetky trénovacie údaje rozdelia na n rovnako veľkých častí. Následne sa natrénuje n modelov. Na trénovanie týchto modelov sa použije množina trénovacích inštancií zostavená z $(n-1)$ cross-validačných častí. Na testovanie modelu sa použije tá cross-validačná časť, ktorá nebola použitá na trénovanie.

Pre všetky naše ďalšie experimenty sme sa rozhodli používať 9-fold cross-validáciu. Rozhodli sme sa tak na základe toho, že sme nastavili prahovú frekvenciu patternov práve na hodnotu 9. To nám umožňuje zostaviť cross-validačné množiny trénovacích údajov tak, že v každej množine bude vždy aspoň jedna inštancia z každého patternu pre dané sloveso.

Ako sme už naznačili v predchádzajúcom odstavci, inštancie do jednotlivých cross-validačných častí sme nevyberali úplne náhodne, ale zvolili sme výber taký, aby sa v každej časti vyskytovali jednotlivé patterny v rovnakom pomere. Vytvorili sme tak stratifikované rozdelenie trénovacích údajov podľa počtu patternov.

Celková úspešnosť modelu na trénovacích inštanciách sa potom spočíta ako priemer úspešností jednotlivých cross-validačných modelov. Zároveň, vďaka dostupným úspešnostiam jednotlivých modelov, je možné spočítať konfidenčný interval pre úspešnosť modelu.

Konfidenčný interval je dobre známy pojem zo štatistiky. Je založený na t-teste s hladinou $\alpha = 0.05$. Znamená to, že s 95% pravdepodobnosťou by mala byť úspešnosť natrénovaných modelov vnútri tohto intervalu.

7.5 Baseline

Základom správnej evaluácie výsledkou strojového učenia je definovanie baseline – základnej úspešnosti, ktorú vieme väčšinou dosiahnuť nejakým triviálnym spôsobom a ktorú sa v ďalších experimentoch snažíme čo možno najviac vylepšiť.

Pre naše experimenty sme si ako baseline zvolili výsledky naivného klasifikátora. Naivný klasifikátor v našom prípade ohodnotí každú testovaciu inštanciu najpravdepodobnejším patternom pre dané sloveso. Najpravdepodobnejší pattern je ten, ktorý sa v trénovacích inštanciách vyskytol najčastejšie.

Okrem samotnej Accuracy, ktorú v celej práci považujeme za hlavný ukazovateľ úspešnosti modelov budeme v celej práci sledovať aj šírku konfidenčných intervalov. Budeme sa snažiť, aby šírka intervalov bola čo možno najmenšia.

V tabuľkách 7.5 a 7.6 prezentujeme dosiahnuté hodnoty Accuracy na jednotlivých slovesách a šírky konfidenčných intervalov. Z výsledkov vyplýva, že najvyššiu Accuracy majú slovesá, ktoré majú zároveň najnižšiu perplexitu. Je to úplne v súlade s princípom naivného klasifikátora. Slovesá s nízkou perplexitou majú medzi patternami jeden pattern dominantný, tj. veľmi frekventovaný. Naivný klasifikátor pri trénovaní zvolil práve ten ako najpravdepodobnejší.

Sloveso	**S.**	**Perplexita**	**Acc.**	**Sloveso**	**S.**	**Perplexita**	**Acc.**
access	C	3.143	47.0 ±1.5	part	C	6.179	43.0 ±1.5
ally	C	3.921	47.6 ±1.2	plough	C	6.859	32.4 ±0.4
arrive	B	2.965	68.0 ±1.4	plug	C	8.653	31.3 ±1.3
breathe	C	5.084	37.7 ±1.1	pour	C	7.901	24.3 ±0.9
claim	A	2.958	67.8 ±0.7	say	A	1.906	85.2 ±0.6
cool	C	5.511	27.3 ±0.9	smash	C	3.967	53.4 ±1.3
crush	C	6.926	29.4 ±1.0	smell	C	5.844	36.3 ±1.0
cry	B	3.532	52.4 ±1.2	steer	C	11.105	20.0 ±1.1
deny	B	5.173	44.4 ±1.2	submit	B	2.627	70.8 ±0.9
enlarge	C	2.290	76.7 ±1.7	swell	C	9.041	21.3 ±0.9
enlist	C	3.493	49.0 ±0.6	tell	A	3.767	65.2 ±0.6
forge	C	7.373	26.3 ±1.0	throw	B	16.918	22.7 ±0.3
furnish	C	4.295	43.7 ±1.6	trouble	C	6.150	44.3 ±1.0
hail	C	2.892	67.4 ±1.8	wake	C	4.765	45.0 ±0.5
halt	C	1.806	83.6 ±1.3	yield	B	7.371	28.3 ±1.0

Tabuľka 7.5: Baseline získaná pomocou naivného klasifikátora pre každé z 30 slovies kolekcie VPS.

Skupina	**Perplexita**	**Acc.**
A	2.320	80.2 ±0.6
B	6.482	50.0 ±1.0
C	5.342	43.9 ±1.1
Celkom	3.044	73.9 ±0.7

Tabuľka 7.6: Baseline získaná pomocou naivného klasifikátora pre skupiny slovies kolekcie VPS.

Kapitola 8

Trénovanie a optimalizácia modelov

V tejto kapitole prezentujeme výsledky trénovania, ladenia a optimalizácie modelov klasifikátorov, ktoré sme v práci vykonali. Okrem samotných číselných výsledkov prezentujeme aj naše názory a komentáre k dosiahnutým výsledkom a prijímame závery, z ktorých vychádzame pri následných experimentoch.

8.1 Ladenie algoritmov strojového učenia

V našej práci sme sa rozhodli experimentovať s nasledujúcimi štyrmi algoritmami strojového učenia:

1. Rozhodovacie stromy (DT)
2. K najbližších susedov (kNN)
3. Podporné vektory (SVM)
4. Adaboost (ADA)

Podrobnosti o týchto algoritmoch sme uviedli v samostatnej kapitole 3. Každý z týchto algoritmov obsahuje možnosť nastavenia určitých parametrov, ktoré môžu výrazne ovplyvniť celkovú úspešnosť modelu. Proces výberu optimálnych parametrov sa nazýva **ladenie modelov** (*tuning*).

V tejto časti práce popisujeme spôsoby, akými sme pre jednotlivé algoritmy hľadali optimálne nastavenia parametrov.

Rozhodovacie stromy (DT)

Vo všetkých experimentoch s rozhodovacími stromami sme používali implementáciu dostupnú v štatistickom systéme `R` [52] konkrétne v knižnici `rpart` [53]. Táto implementácia umožňuje nastavovať niekoľko rôznych parametrov. Rozhodli sme sa ladiť dva z týchto parametrov – parameter `cp` a `minsplit`.

Parameter `cp` (*complex parameter*) je komplexný parameter, ktorý ovplyvňuje výslednú podobu rozhodovacieho stromu. Parameter `minsplit` určuje počet inštancií, ktoré musia padnúť do uzla stromu na to, aby sa tento uzol mohol stať rozhodovacím a ďalej rozdeliť inštancie.

Pre potreby našich experimentov sme vytvorili skript v jazyku `R`, ktorý postupne trénoval modely s rôznymi hodnotami parametra `cp`. Pre každú z týchto hodnôt skript natrénoval a otestoval 9 cross-validačných modelov. Za najlepšiu hodnotu parametra

`cp` sme prehlásili takú hodnotu, ktorá bola najmenšia, a pri ktorej bola priemerná úspešnosť cez všetky cykly cross-validácie najvyššia.

Po získaní najlepšieho parametra `cp` sme pristúpili k optimalizácii parametra `minsplit`. Opäť sme trénovali a testovali modely s rôznymi hodnotami `minsplit`, avšak už s nastaveným najlepším `cp`. Za najlepšiu hodnotu sme prehlásili také nastavenie, pri ktorom bola priemerná úspešnosť cross-validačných modelov najvyššia.

K najbližších susedov (kNN)

V experimentoch sme používali implementáciu tohto algoritmu dostupnú v `R`, konkrétne v knižnici `e1071` [54].

Algoritmus k najbližších susedov umožnuje optimalizovať len parameter `k`, ktorý definuje, koľko podobnejších inštancií – susedov – sa má brať do úvahy pri klasifikovaní testovacej inštancie.

Okrem toho je možné zmeniť úspešnosť algoritmu aj **normalizáciou** (*scaling*) vektorov rysov. Normalizácia spočíva v tom, že sa hodnoty rysov v jednotlivých vektoroch matematicky znormalizujú tak, aby mal každý rys rovnakú váhu. Rovnaká váha rysov je dôležitá predovšetkým v geometrických metódach, v ktorých sa vzdialenosti medzi vektormi počítajú pomocou euklidovskej metriky.

Pri hľadaní optimálneho modelu sme natrénovali a otestovali modely s rôznymi hodnotami parametra `k` a to bez použitia normalizácie, ale aj s použitím normalizácie.

Za najlepší sme prehlásili model, ktorý dosiahol najvyššiu priemernú úspešnosť v priebehu cross-validácie.

Podporné vektory (SVM)

Experimentovali sme s implementáciou algoritmu v systéme `R`. Algoritmus je implementovaný opäť v knižnici `e1071`.

Implementácia umožňuje nastaviť niekoľko dôležitých parametrov, ktoré môžu aj veľmi výrazne zmeniť úspešnosť natrénovaných modelov. Prvým experimentom, ktorým sme sa snažili zvýšiť úspešnosť, bolo zvolenie vhodného typu kernelovej funkcie (parameter `kernel`). Implementácia SVM umožňuje vybrať si niektorý z týchto typov kernelových funkcií:

1. Lineárny typ
2. Polynomiálny typ
3. Radiálny typ

Pre každý z týchto typov sú následne definované ďalšie špeciálne parametre. Pri ladení sme postupovali tak, že najskôr sme vybrali najúspešnejšie nastavenie parametra `kernel`. Následne sme optimalizovali ďalšie parametre pre zvolený `kernel`:

1. Pre lineárny kernel už nie je k dispozícii žiaden ďalší parameter.
2. Pre polynomiálny kernel sme optimalizovali parameter `degree`, ktorým sa definuje maximálny možný stupeň polynómu danej kernelovej funkcie. Po zvolení najúspešnejšieho stupňa sme ešte optimalizovali parameter `gamma` a zvolili takú hodnotu, na ktorej model dosiahol najvyššiu priemernú úspešnosť.
3. Pre radiálny kernel je možné optimalizovať len parameter `gamma`. Zvolili sme takú hodnotu, pri ktorej model dosiahol najvyššiu priemernú úspešnosť.

Po zvolení najlepšiej kernelovej funkcie a s tým súvisiacich parametrov sme ešte ladili posledný dostupný parameter `cost`. Ten umožňuje nastaviť penalizáciu za inštancie, ktoré sú zaradené do zlej nadroviny (tj. môžu byť klasifikované chybne). Tým umožňujeme algoritmu vypočítať jednoduchší model nadroviny, s predpokladom, že bude všeobecnejší než zložitejší model a bude tak viac robustný pri klasifikovaní nevidených inštancií.

Adaboost (ADA)

Použili sme implementáiu Adaboostu v systéme `R`, konkrétne v knižnici `adabag` [55]. Táto implementácia Adaboostu vytvára 100 modelov rozhodovacích stromov. To umožňuje optimalizovať Adaboost pomocou rovnakých parametrov, ako rozhodovacie stromy.

Pri optimalizácii parametrov `cp` a `minsplit` sme použili rovnaký postup ako pri optimalizácii úspešnosti rozhodovacích stromov. Najskôr sme vybrali najlepšiu hodnotu parametra `cp`, následne sme túto hodnotu použili pre nájdenie najlepšej hodnoty parametra `minsplit`.

8.2 Selekcia morfo-syntaktických rysov

Jedným z kľúčových experimentov tejto práce bola snaha optimalizovať množinu morfo-syntaktických rysov. Táto množina bola navrhnutá ľudmi intuitívne. Definovali sme také rysy, ktoré sa ľudským expertom zdali byť najužitočnejšie. Cieľom nasledujúcich experimentov bolo predovšetkým zistiť

- ktoré rysy sú skutočne užitočné pre algoritmy strojového učenia; a
- či nie je možné odstrániť z množiny rysov nejaké rysy bez zníženia úspešnosti.

V tejto časti práce popisujeme experimenty, ktorými sme sa snažili vytvoriť optimálnu množinu morfo-syntaktických rysov, s ktorou by modely klasifikátorov dosahovali najvyššie úspešnosti.

Formát tabuliek s výsledkami modelov

Výsledky všetkých experimentov, v ktorých sme trénovali modely klasifikátorov pomocou štyroch algoritmov strojového učenia a ladili ich spôsobom popísaným vyššie, budeme prezentovať v tabuľkách, ktorých formát sa riadi naledujúcimi pravidlami:

- V texte práce uvádzame vždy len súhrné výsledky pre jednotlivé skupiny. Podrobné výsledky pre všetky slovesá uvádzame v dodatku D. V komentári ku každej tabuľke uvádzame odkaz na tabuľku s podrobnými výsledkami.
- V tabuľkách uvádzame najlepšie priemerné hodnoty Accuracy, ktoré sme v cross-validáci dosiahli pomocou daného algoritmu. Najlepšie hodnoty boli získané ladením.
- Okrem hodnôt Accuracy uvádzame za znakom $\pm$ polomer konfidenčného intervalu. Výpočet konfidenčného intervalu je založený na štandardnom t-teste pri hladine testu $\alpha = 5\%$. Výpočet je prevedený pomocou deviatich hodnôt Accuracy získaných v cross-validácii.
- V komentári k tabuľke vždy uvádzame, ktorý zo štyroch algoritmov považujeme za najlepší.
- V stĺpcoch **Zlepšenie** uvádzame porovnanie najlepšieho algoritmu s Baseline.

- V stĺpci **r** uvádzame percentuálne body, o ktoré sa zvýšila Accuracy najlepšieho modelu (Acc_M) oproti Baseline (Acc_B): $Acc_M - Acc_B$.
- V stĺpci **%** uvádzame percentuálne zlepšenie oproti Baseline: $(Acc_M/Acc_B) - 1$.
- V stĺpci **ERD** uvádzame, koľko percent chybných klasifikácií sa podarilo odstrániť oproti Baseline:

$$1 - \frac{1 - Acc_M}{1 - Acc_B}$$

8.2.1 Modely využívajúce celú množinu rysov

Ešte pred samotným optimalizovaním morfo-syntaktickej množiny rysov sme natrénovali a vyladili modely klasifikátorov, ktoré používajú celú množinu morfo-syntaktických rysov. Tento experiment sme pomenovali **Default-FS 149**. Výsledky experimentu prezentujeme v tabuľke 8.1.

Za najlepší algoritmus v tomto experimente považujeme SVM, hoci žiaden algoritmus nemôžeme prehlásiť za najlepší aj z hľadiska štatistickej signifikantnosti. Pre väčšinu slovies úspešnosť vzrástla minimálne o 20 percentuálnych bodov. Naopak, úspešnosť zostala približne rovnaká pre slovesá, ktoré dosahovali vysokú úspešnosť už pri použití naivného klasifikátora. Môžeme pozorovať jasnú koreláciu medzi veľkosťou zlepšenia a perplexitou slovesa. Čím bola perplexita nižšia, tým menšie zlepšenie sa nám podarilo dosiahnuť.

					Zlepšenie		
Skupina	**DT**	**kNN**	**SVM**	**ADA**	**r**	**%**	**ERD**
A	84.8 ±1.2	85.3 ±2.1	86.9 ±2.2	86.3 ±2.5	6.7	9.0	32.8
B	60.9 ±5.5	59.5 ±4.5	61.3 ±3.7	62.2 ±4.0	11.4	32.6	22.2
C	63.9 ±4.4	62.0 ±4.2	66.1 ±4.3	65.3 ±4.7	22.4	67.5	38.7
Celkom	80.4 ±2.0	80.5 ±2.5	82.3 ±2.5	81.8 ±2.8	8.4	16.0	31.9

Tabuľka 8.1: Výsledky experimentu **Default-FS 149**. Modely klasifikátorov boli natrénované pomocou všetkých dostupných morfologicko-syntaktických rysov. Najlepším algoritmom je SVM, hoci medzi výsledkami algoritmov neexistuje štatistiky signifikantný rozdiel. Podrobné výsledky pre všetky slovesá uvádzame v dodatku D.10 na strane 108.

8.2.2 Uporiadanie rysov podľa úspešnosti

V snahe optimalizovať množinu morfo-syntaktických rysov sme sa znažili zistiť, ktoré rysy majú najväčší podiel na správnom klasifikovaní inštancií a naopak, ktoré rysy klasifikátory pri trénovaní a klasifikovaní vôbec nepoužívajú.

Výsledkom experimentov bolo zostavenie niekoľkých usporiadaní rysov, ktoré určitým spôsobom vyjadrovali dôležitosť rysu pri riešení klasifikačnej úlohy.

Zisťovali sme

1. akú úspešnosť majú rozhodovacie stromy pri klasifikovaní každého patternu osobitne a s použitím len jedného rysu;
2. aké rysy sa používajú v rozhodovacích stromoch pri klasifikovaní každého patternu osobitne; a
3. aké rysy vyberie hladový algoritmus, ktorý sa snaží maximalizovať Accuracy klasifikátorov.

Binárne klasifikátory s použitím jedného rysu

V tomto experimente sme sa namiesto jedného klasifikátora pre každé sloveso rozhodli vytvoriť samostatné klasifikátory pre každý pattern každého slovesa. Klasifikátory jednotlivých patternov boli binárne – ich úlohou bolo rozhodnúť, či sa jedná o daný pattern alebo nie.

Navyše, každý klasifikátor mohol pri svojom rozhodovaní použiť len jediný rys. Výsledkom experimentu bolo natrénovanie $|MS| \cdot |P|$ klasifikátorov, kde MS je množina morfologicko-syntaktických rysov a P je množina všetkých patternov všetkých 30 slovies.

Pre každý z týchto klasifikátorov sme cross-validáciou získali priemernú Accuracy. Túto Accuracy budeme ďalej označovať ako $A_{f,p}$, kde f je rys, ktorý mohol klasifikátor použiť a p je pattern, ktorý klasifikátor určoval.

Získali sme tak sadu úspešností jednotlivých klasifikátorov. Túto sadu použijeme na zostavenie rebríčka najúspešnejších rysov. Formálne môžeme rebríček rysov pre daný pattern p definovať pomocou funkcie o_p, ktorá každému rysu priradí prirodzené číslo, vyjadrujúce pozíciu rysu v rebríčku.

Pre každý pattern p nech funkcia $o_p : MS \to \{1, 2, ..., |MS|\}$ zoradzuje rysy zostupne podľa hodnôt $\{A_{f,p}; f \in MS\}$. Následne môžme definovať číslo $N_{f,v}$, ktoré bude vyjadrovať úspešnosť rysu f pre sloveso v nasledujúcim spôsobom:

$$N_{f,v} = \sum_{p \in P_v} \frac{1}{o_p(f)} A_{f,p},$$

kde P_v je množina patternov definovaných pre sloveso v.

Analogicky môžme definovať zoradzovaciu funkciu $r_v : MS \to \{1, 2, ..., |MS|\}$, ktorá zoradí zostupne rysy pre jednotlivé slovesá podľa hodnôt $\{N_{f,v}; f \in MS\}$. Výsledné číslo U_f bude definované pre všetky rysy f nasledujúcim spôsobom:

$$U_f = \sum_{v \in V} \frac{1}{r_v(f)} N_{f,v},$$

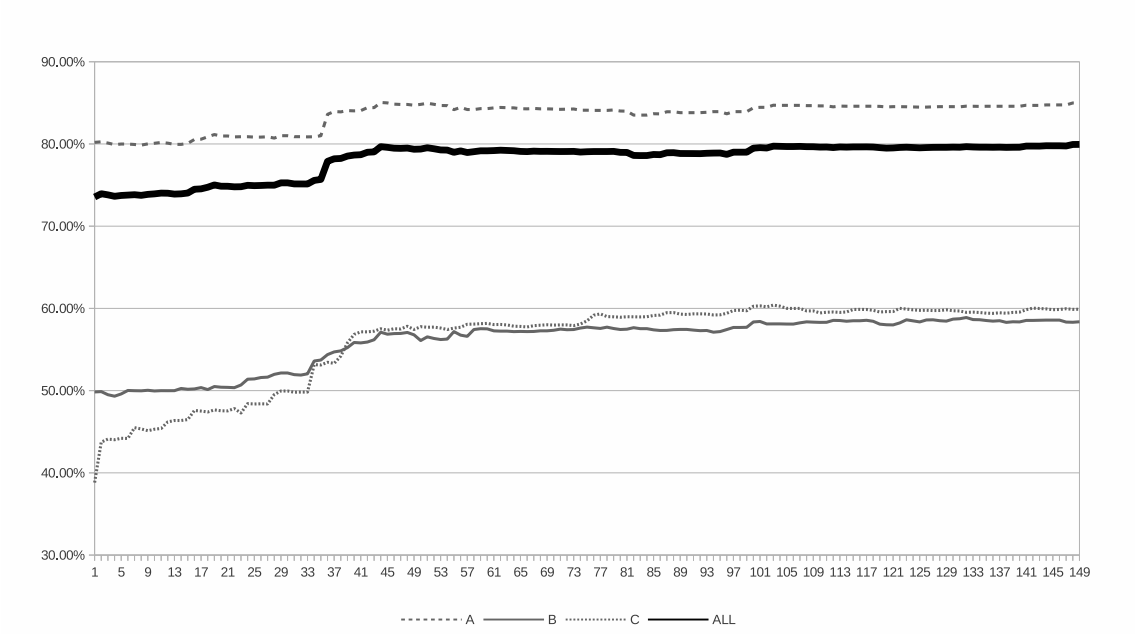
kde V je množina všetkých slovies.

Jednotlivé rysy zoradené zostupne podľa hodnôt U_f budú tvoriť výsledný rebríček rysov. Tento rebríček nazývame **Rebríček A** a prezentujeme ho v dodatku D.11 na strane 109.

Po získaní rebríčka sme prvých 20 rysov použili na natrénovanie modelov klasifikátorov, ktoré sme následne vyladili. Tento experiment sme pomenovali **A20**. Výsledky prezentujeme v tabuľke 8.2.

					Zlepšenie		
Skupina	**DT**	**kNN**	**SVM**	**ADA**	**r**	**%**	**ERD**
A	82.3 ±1.5	81.8 ±1.3	81.5 ±0.9	82.3 ±1.5	1.4	2.1	4.1
B	51.7 ±1.7	52.2 ±3.1	52.2 ±3.2	52.0 ±2.6	2.2	7.5	3.7
C	49.1 ±3.3	48.7 ±3.7	49.4 ±3.7	49.1 ±3.8	5.6	17.4	9.1
Celkom	76.2 ±1.7	75.8 ±1.7	75.6 ±1.4	76.2 ±1.8	1.8	3.8	4.4

Tabuľka 8.2: Výsledky experimentu **A20**. Modely klasifikátorov boli natrénované pomocou 20 najlepších rysov podľa rebríčka A. Najlepším algoritmom je SVM, hoci medzi výsledkami algoritmov neexistuje štatistiky signifikantný rozdiel. Podrobné výsledky pre všetky slovesá uvádzame v dodatku D.12 na strane 110.



Obr. 8.1: Vývoj Accuracy pri použití SVM natrénovaných na prvých n rysoch z rebríčka A.

Celkovo môžme hodnotiť výkonnosť tohto modelu ako málo uspokojivú. Väčšina modelov pre jednotlivé slovesá sa signifikatne zhoršila oproti modelom natrénovaným na celej sade morfologicko-syntaktických rysov.

Chceme upozorniť na zaujímavý jav – napriek celkovému zníženiu úspešnosti môžme pozorovať zúženie konfidenčných intervalov. Znamená to, že výsledky jednotlivých modelov vytvorených v cross-validácii sa od seba líšili menej ako pri modeloch natrénovaných na celej množine rysov.

Rozhodnutie, že z rebríčka A zvolíme prvých 20 rysov nebolo podložené žiadnym experimentom. Chceli sme vedieť, aká úspešnosť bude dosiahnutá, ak zvolíme prvých 10, 30 alebo 40 rysov. Keďže trénovanie a ladenie všetkých možných veľkostí množiny rysov by bolo náročné, využili sme naše pozorovanie, že ladenie jednotlivých modelov všeobecne zvyšuje úspešnosť len o 1 až 2 percentuálne body. Rozhodli sme sa preto modely s rôznymi veľkosťami množiny rysov neladiť, čím sme časovú náročnosť experimentu znížili natoľko, že sme mohli natrénovať a otestovať modely pre všetky veľkosti množiny rysov.

Z rebríčka A sme postupne vytvárali rôzne veľké množiny rysov, pričom platilo, že sme do n-prvkovej množiny vkladali vždy prvých n rysov z rebríčka. Trénovanie prebehlo so všetkými štyrmi algoritmami. Výsledky prezentované graficky na obrázku 8.1 ukazujú vývoj Accuracy slovies, troch skupín a celkovú pre algoritmus SVM.

Z prezentovaného grafu vyplýva, že pridávaním rysov sa Accuracy postupne zvyšuje. Pre skupiny A a B nastáva maximum približne pri počte rysov 40. Pre skupinu C je to až niekde okolo počtu 100.

Na základe prezentovaných výsledkov si myslíme, že rebríček A prepredstavuje ideálne zoradenie rysov. Pristúpili sme k ďalšiemu experimentu, ktorého výsledkom bol odlišný rebríček.

Binárne klasifikátory s použitím všetkých rysov

V tomto experimente sme opäť vytvorili binárne klasifikátory pre každý jednotlivý pattern. Opäť sme použili algoritmus rozhodovacích stromov, ktorým sme ale tentoraz umožnili pracovať so všetkými dostupnými rysmi. Zaujímalo nás, ktoré rysy sa v rozhodovacom strome uplatnia a predovšetkým nás zaujímalo, v akej hĺbke rozhodovacieho stromu sa ten-ktorý rys bude nachádzať.

Pod pojmom hĺbka rysu v rozhodovacom strome rozumieme dĺžku cesty od koreňa k rozhodovaciemu uzlu, v ktorom sa používa daný rys.

Výsledkom experimentu bolo natrénovanie $9 \cdot |MS| \cdot |P|$ klasifikátorov, kde 9 je počet cross-validačných cyklov.

Pre každý z týchto klasifikátorov sme získali hĺbku, v ktorej sa daný rys f vyskytoval v rozhodovacom strome pre klasifikátor patternu p v cross-validačnom cykle $c = 1, ..., 9$. Túto hĺbku budeme ďalej označovať ako $D_{f,p,c}$.

Pomocou hĺbky D sme definovali váhu $W_{f,p}$ rysu f pre daný pattern p predpisom

$$W_{f,p} = \frac{1}{9} \sum_{c=1}^{9} 2^{-D_{f,p,c}}$$

Získali sme tak ďalšiu sadu úspešností jednotlivých klasifikátorov. Pomocou tejto sady čísel sme vytvorili ďalší rebríček rysov – **Rebríček W**. Postupovali sme analogicky ako v prípade rebríčka A, kde sme namiesto sady $A_{f,p}$ použili sadu $W_{f,p}$.

Myšlienkou sledovať hĺbku jednotlivých rysov v rozhodovacích stromoch sme sa inšpirovali v práci [37]. Rebríček W prezentujeme v dodatku D.13, na strane 111. Rebríček neobsahuje všetkých 149 morfologicko-syntaktických rysov. Rysy, ktoré sa v rebríčku nenachádzajú, sa nevyskytli ani v jednom prípade v skúmaných rozhodovacích stromoch.

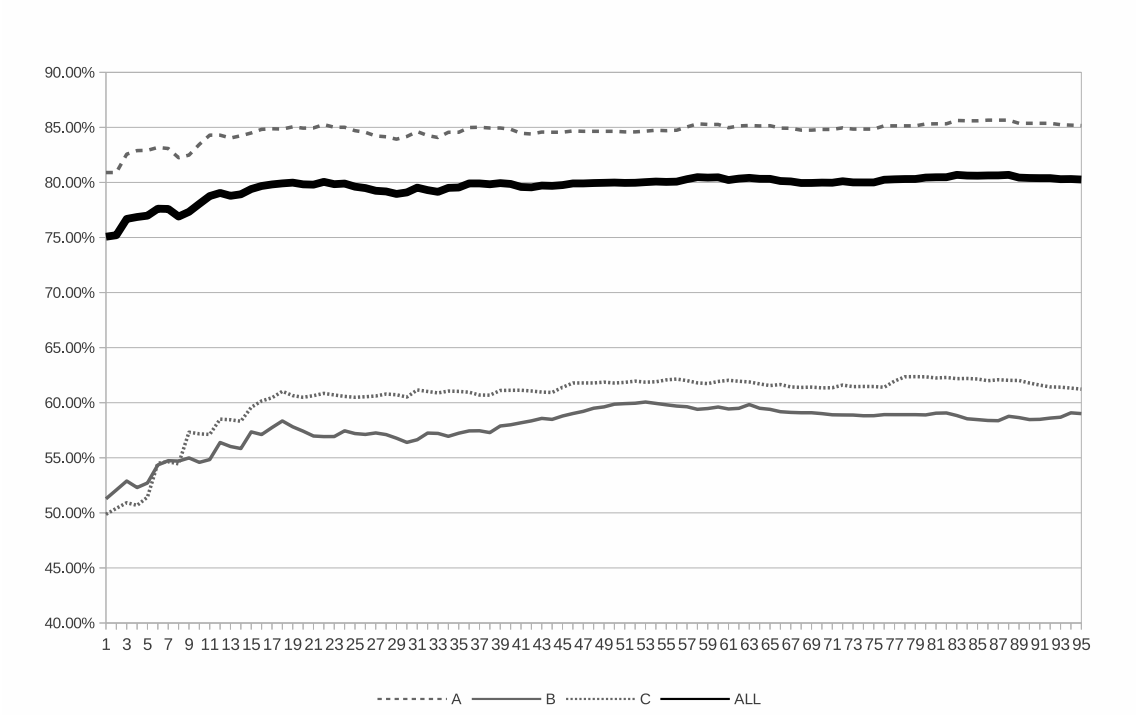
Analogicky, s predchádzajúcim rebríčkom, sme opäť vybrali 20 najlepších rysov a použili ich na natrénovanie modelov klasifikátorov. Tento experiment sme pomenovali **W20**. Výsledky prezentujeme v tabuľke 8.3.

					Zlepšenie		
Skupina	**DT**	**kNN**	**SVM**	**ADA**	**r**	**%**	**ERD**
A	84.6 ±2.1	84.5 ±2.3	85.7 ±2.6	84.6 ±1.8	5.5	7.4	27.7
B	56.3 ±3.6	56.1 ±4.0	58.2 ±3.8	56.4 ±4.4	7.5	18.6	15.9
C	61.2 ±4.1	60.5 ±4.7	63.4 ±4.4	62.3 ±4.3	19.7	58.7	33.5
Celkom	79.4 ±2.4	79.3 ±2.7	80.7 ±2.9	79.6 ±2.3	6.8	12.4	26.6

Tabuľka 8.3: Výsledky experimentu **W20**. Modely klasifikátorov boli natrénované pomocou 20 najlepších rysov podľa rebríčka W. Najlepším algoritmom je SVM, hoci medzi výsledkami algoritmov neexistuje štatistiky signifikantný rozdiel. Podrobné výsledky pre všetky slovesá uvádzame v dodatku D.14 na strane 112.

Porovnaním výsledkov s výsledkami dosiahnutými pomocou celej množiny morfologicko-syntaktických rysov sme zistili, že zmeny v Accuracy nie sú pre jednotlivé slovesá štatisticky signifikantné. Zvýšenie Accuracy sme zaznamenali u šiestich slovies.

Porovnávali sme tiež šírky konfidenčných intervalov a konštatujeme celkové zväčšenie šírky, čo považujeme za negatívny jav. Na základe uvedeného si myslíme, že prvých 20 rysov podľa ich výskytu v rozhodovacích stromoch nie je dobrou náhradou za celú množinu morfologicko-syntaktických rysov.



Obr. 8.2: Vývoj Accuracy pri použití SVM natrénovaných na prvých n rysoch z rebríčka W.

Podobne, ako s použitím rebríčka A sme aj pre poradie definované rebríčkom W natrénovali a evaluovali modely klasifikátorov používajúce rôzny počet rysov. Výsledný vývoj Accuracy pre algortimus SVM prezentujeme na obrázku 8.2.

Porovnaním obrázkov 8.1 a 8.2 sme zistli, že použitím rebríčka W je maximálna úspešnosť dosiahnutá použitím menšieho počtu rysov. Pre skupiny A a C je to približne pri počte 20 a pre skupinu B pri počte 50 rysov.

Na základe tohto experimentu si myslíme, že rebríček W predstavuje lepšie zoradenie rysov, než rebríček A.

Hladový algoritmus pre výber rysov s binárnymi klasifikátormi

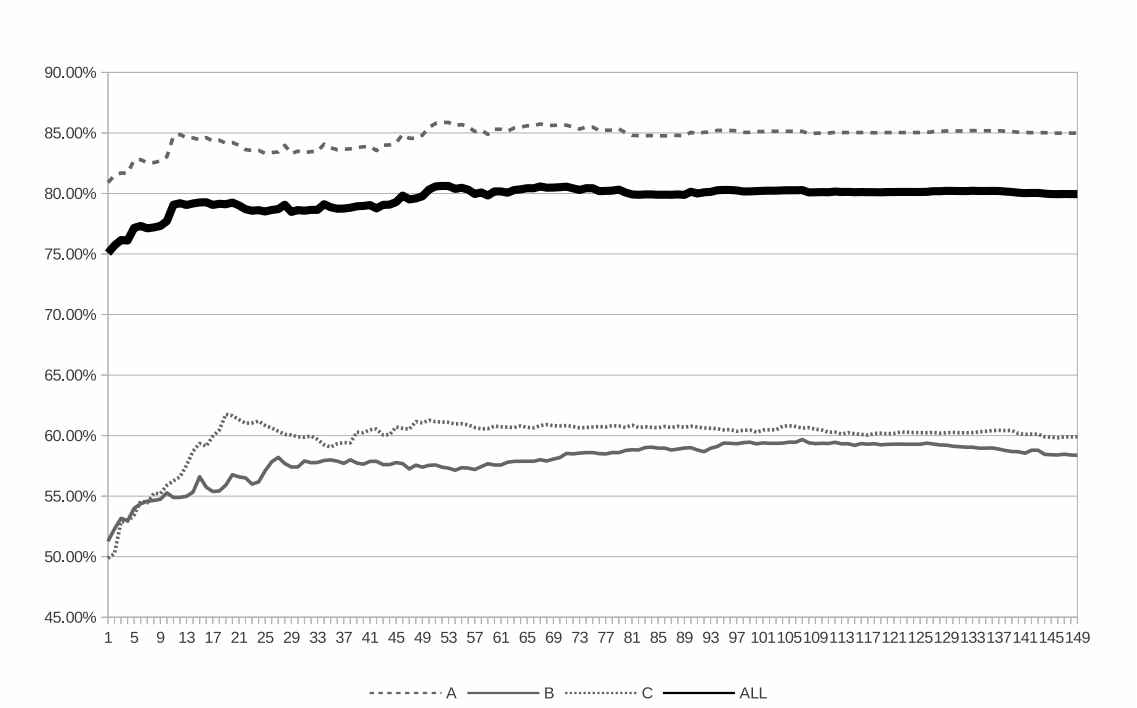
V tomto experimente sme opäť vytvárali binárne klasifikátory pre každý pattern každého slovesa. Opäť sme použili algoritmus rozhodovacích stromov a implementovali sme dobre známy hladový (*greedy*) algoritmus.

Definovali sme množinu F, ako množinu rysov, ktoré môže klasifikátor použiť pri trénovaní. Ďalej sme definovali množinu dostupných rysov DF, ktorá obsahovala všetky morfologicko-syntaktické rysy, ktoré ešte neboli zahrnuté v množine F.

Na začiatku algoritmu sme začali s prázdnou množinou F a množinou DF, ktorá obsahovala všetky dostupné rysy. V každom kole sme našli taký rys $f \in DF$, ktorý maximalizoval Accuracy binárneho klasifikátora, ktorý bol natrénovaný na množine rysov $F \cup \{f\}$. Takýto najúspešnejší rys potom sme zaradili do množiny F a odobrali z množiny DF. Takýmto spôsobom sme pokračovali, kým nebola množina DF prázdna.

Pre každý klasifikátor patternu p, ktorý sme obohatili o rys f sme si zapamätali Accuracy, ktorú sme označili ako $G_{f,p}$. Získali sme tak ďalšiu sadu úspešností jednotlivých klasifikátorov. Túto sadu sme použili na vytvorenie **rebríčka G**, podobne, ako v predchádzajúcich dvoch experimentoch.

Rebríček G prezentujeme v dodatku D.15, na strane 113.



Obr. 8.3: Vývoj Accuracy pri použití SVM natrénovaných na prvých n rysoch z rebríčka G.

Podobne, ako v prípade predchádzajúcich rebríčkov sme aj rebríček G použili na natrénovanie modelov s rôznym počtom rysov. Výsledok experimentu s algoritmom SVM prezentujeme na obrázku 8.3.

Z uvedeného grafu vyplýva, že maximálne hodnoty Accuracy dosahuje pre skupiny A a C už pri použití približne 15 až 20 rysov. V porovnaní s grafom pre rebríček W sa dosiahnutie celkovej Accuracy podarilo s použitím ešte menšieho počtu rysov. Pre skupinu B bolo ale dosiahnutie maxima možné až s použitím viac ako stovky rysov.

Na základe uvedeného si myslíme, že rebríčky W a G sú približne rovnako dobré a spoločne sú lepšie, než rebríček A.

Hladový algoritmus pre výber rysov s klasifikátorom pre všetky patterny

V tomto experimente sme použili hladový algoritmus na klasifikátory, ktoré klasifikujú všetky patterny daného slovesa. Algoritmus pracoval rovnako, ako v predchádzajúcom prípade. Jediným rozdielom bolo, že algoritmus nemal za úlohu postupne pridať do množiny F všetky rysy, ale s pridávaním rysov skončil vo chvíli, keď žiaden rys nezvýšil Accuracy nad Accuracy dosiahnutú v predošlom kole. Algoritmus teda skončil pridávanie po dosiahnutí prvého (lokálneho) maxima.

Výsledkom experimentu bola množina rysov, ktoré sú najúspešnejšie pre dané sloveso. Prehľad týchto rysov uvádzame v dodatku D.16, na strane 114.

Uvažujme pre každé sloveso funkciu $w_v : |MS| \to \mathbb{Q}$, ktorá priradí váhu každému rysu podľa toho, v akom poradí ho hladový algoritmus vybral do množiny F, resp. priradí 0, ak algoritmus rys nepoužil. Váhu w_v sme pre použité rysy vypočítali predpisom:

$$w_v(f) = \frac{1}{o_v(f)},$$

kde o_v je spomínané poradie pridania do množiny F. Na základe týchto váh sme potom

vypočítali váhu rysu globálne, pre všetky slovesá predpisom:

$$w_(f) = \sum_{v \in V} w_v(f).$$

Získali sme tak ďalší rebríček 58 rysov, ktorý nazývame **Best58**. Rebríček uvádzame v dodatku D.17, na strane 115.

Zaujímalo nás, na koľko je rebríček Best58 podobný s rebríčkom G. Zistili sme, že na prvých 58 miestach rebríčka G sa nachádza až 40 rysov z rebríčka Best58, čo predstavuje takmer 70% podobnosť.

Na základe tejto podobnosti sme sa rozhodli použiť všetkých 58 rysov získaných v tomto experimente na natrénovanie modelov klasifikátorov. Tento experiment sme pomenovali **Best58**. Výsledky prezentujeme v tabuľke 8.4.

					Zlepšenie		
Skupina	**DT**	**kNN**	**SVM**	**ADA**	**r**	**%**	**ERD**
A	85.6 ±2.4	85.1 ±1.7	87.0 ±2.4	86.0 ±2.2	6.8	9.1	34.3
B	61.2 ±4.7	59.5 ±4.3	62.1 ±3.8	62.1 ±4.1	12.1	34.9	23.6
C	63.6 ±4.1	62.6 ±4.3	66.0 ±4.8	64.9 ±4.7	22.2	66.5	38.2
Celkom	81.0 ±2.8	80.4 ±2.2	82.4 ±2.8	81.6 ±2.6	8.6	16.3	33.3

Tabuľka 8.4: Výsledky experimentu **Best58**. Modely klasifikátorov boli natrénované pomocou 58 najlepších rysov podľa rebríčka Best58. Najlepším algoritmom je SVM, hoci medzi výsledkami algoritmov neexistuje štatistiky signifikantný rozdiel. Podrobné výsledky pre všetky slovesá uvádzame v dodatku D.18 na strane 116.

Výsledky modelov sú z hľadiska štatistikej signifikantnosti rovnako dobré, ako výsledky modelov natrénovaných nad celou množinou morfologicko-syntaktických rysov. Pre 15 slovies navyše nastalo zlepšenie v Accuracy, ktoré prispelo k tomu, že v skupine slovies A a B sme dosiahli zatiaľ najlepšie výsledky Accuracy.

Sledovali sme tiež zmenu konfidenčných intervalov jednotlivých výsledkov. Musíme konštatovať, že takmer pre všetky slovesá došlo k rozšíreniu intervalov. Globálne sa ale polomer intervalov zväčšil maximálne o 0,5 percentného bodu.

Na základe týchto skutočností sme sa rozhodli prehlásiť model natrénovaný na 58 rysoch za zatiaľ najlepší model.

Posledným experimentom, ktorý sme vykonali v rámci snahy optimalizovať množinu morfo-syntaktických rysov, bolo použitie hladového algoritmu na množinu rysov Best58. Namiesto pridávania rysov ale tentoraz hladový algoritmus vyberal a odstraňoval rysy, ktoré najviac znižovali Accuracy pre jednotlivé slovesá. Je zaujímavé, že pre každé sloveso sa našiel minimálne jeden rys, ktorého odobratím sa zvýšila úspešnosť modelu. Po získaní redukovaného zoznamu rysov sme pre každé sloveso natrénovali a vyladili modely s použitím všetkých štyroch algoritmov strojového učenia. Výsledky tohto experimentu priniesli mierne zvýšenie Accuracy oproti experimentu Best58, nešlo však o štatistiky signifikantné zlepšenie. Navyše, pre každé sloveso bola použitá mierne odlišná sada rysov.

Rozhodli sme sa preto za najlepšiu množinu morfo-syntaktických rysov prehlásiť množinu Best58. Pôvodnú množinu morfo-syntaktických rysov, ktorá obsahovala 149 rysov sme tak dokázali zredukovať takmer na tretinu a to bez zníženia úspešnosti, dokonca s miernym zlepšením (ktoré však nebolo štatisticky signifikantné).

8.3 Selekcia sémantických rysov

Vďaka experimentom popísaným v predchádzajúcej sekcii sme vybrali sadu 58 morfo-syntaktických rysov, ktoré sme označili ako **Best58**. K tejto množine rysov sme sa snažili pripojiť sémantické rysy v snahe zvýšiť celkovú úspešnosť klasifikátorov.

V časti 6.2 sme definovali 3 sady sémantických rysov. Sadu 3 rysov z rozpoznávača menných entít sme označili ako **NER**. Sadu 44 rysov zostavených z 22 hlavných zastrešujúcich prototypov sme označili ako **MU44** (*Major umbrellas 44*). Sadu 124 rysov zostavených zo 62 zastrešujúcich prototypov sme označili ako **AU124** (*All umbrellas 124*).

8.3.1 Modely využívajúce celú množinu rysov

Prvé experimenty, ktoré sme implementovali, sú analogické experimentu s celou sadou morfo-syntaktických rysov popísanému v časti 8.2.1.

V prvom experimente sme natrénovali a vyladili modely klasifikátorov na celej množine morfo-syntaktických rysov (použitých v experimente Default-FS) doplnenou o množinu NER. Výsledky prezentujeme v tabuľke 8.5. Experiment sme nazvali **Default-FS+NER**

					Zlepšenie		
Skupina	**DT**	**kNN**	**SVM**	**ADA**	**r**	**%**	**ERD**
A	84.6 ±1.2	84.9 ±2.3	86.7 ±2.1	86.2 ±2.3	6.6	8.8	31.8
B	60.6 ±5.7	59.7 ±5.1	61.4 ±3.8	61.4 ±3.6	11.4	33.4	22.0
C	64.1 ±4.4	61.7 ±4.5	66.0 ±4.0	65.5 ±4.4	22.3	67.3	38.3
Celkom	80.2 ±2.0	80.2 ±2.8	82.1 ±2.5	81.7 ±2.6	8.3	16.0	31.1

Tabuľka 8.5: Výsledky experimentu **Default-FS+NER**. Modely klasifikátorov boli natrénované pomocou všetkých morfo-syntaktických rysov a troch rysov z rozpoznávača menných entít. Najlepším algoritmom je SVM, hoci medzi výsledkami algoritmov neexistuje štatistiky signifikantný rozdiel. Podrobné výsledky pre všetky slovesá uvádzame v dodatku D.19 na strane 117.

Po pridaní troch nových rysov sme dosiahnuté výsledky porovnali s modelom Best58. Musíme konštatotať, že rozdiel medzi úspešnosťami modelov nie je štatisticky signifikantný. Napriek tomu, vo väčšine prípadov nastalo zhoršenie priemernej Accuracy. Najúspešnejším algoritmom sa stal opäť SVM, nejedená sa však o štatisticky signifikantné víťazstvo. Použitím tohto modelu sa zlepšila Accuracy pre 8 slovies. Trom z týchto slovies sa ale zväčšil konfidenčný interval, čo nepovažujeme sa zlepšenie v pravom zmysle slova. Zlepšenie, vrátane zúženia konfidenčného intervalu nastalo teda len u 5 slovies.

Na základe týchto nepresvedčivých výsledkov sme sa rozhodli, že v ďalších experimentoch nebudeme rysy NER používať.

Druhým experimentom bolo natrénovanie a vyladenie modelov pomocou rysov Best58 doplnenou o množinu MU44. Výsledky prezentujeme v tabuľke 8.6. Experiment sme nazvali **Best58+MU44**.

Ak výsledok experimentu porovnáme s experimentom **Best58**, môžeme vidieť, že nastalo mierne zlepšenie v Accuracy o 1 až 2 percentuálne body. Toto zlepšenie bohužiaľ nie je štatisticky signifikantné. Za pozitívne považujeme, že zlepšenie sme dosiahli pri zachovaní rovnakej šírky konfidenčného intervalu.

Posledným experimentom pred začatím optimalizácie sémantických rysov bolo použitie množiny **Best58** a **AU124**. Výsledky prezentujeme v tabuľke 8.7.

					Zlepšenie		
Skupina	**DT**	**kNN**	**SVM**	**ADA**	**r**	**%**	**ERD**
A	86.4 ±2.0	84.9 ±4.2	87.9 ±2.1	86.8 ±2.5	7.8	10.5	37.0
B	62.5 ±5.4	57.0 ±5.2	65.7 ±3.8	63.4 ±4.0	15.8	45.0	31.5
C	65.2 ±4.3	61.6 ±4.8	68.4 ±4.5	67.0 ±4.7	24.7	74.6	42.1
Celkom	82.0 ±2.6	79.9 ±4.4	83.8 ±2.5	82.5 ±2.8	10.0	19.3	36.7

Tabuľka 8.6: Výsledky experimentu **Best58+MU44**. Modely klasifikátorov boli natrénované pomocou 58 morfo-syntaktických rysov a 44 rysov z množiny MU44. Najlepším algoritmom je SVM, hoci medzi výsledkami algoritmov neexistuje štatistiky signifikantný rozdiel. Podrobné výsledky pre všetky slovesá uvádzame v dodatku D.20 na strane 118.

					Zlepšenie		
Skupina	**DT**	**kNN**	**SVM**	**ADA**	**r**	**%**	**ERD**
A	86.3 ±2.1	85.5 ±1.9	87.7 ±2.2	86.8 ±2.5	7.5	10.2	35.8
B	62.9 ±5.0	61.1 ±4.5	65.3 ±4.8	63.4 ±4.1	15.4	43.8	30.7
C	65.4 ±4.4	64.5 ±4.7	68.5 ±4.4	67.0 ±4.8	24.8	75.1	42.1
Celkom	82.0 ±2.6	81.0 ±2.4	83.6 ±2.7	82.5 ±2.8	9.7	18.9	35.6

Tabuľka 8.7: Výsledky experimentu **Best58+AU124**. Modely klasifikátorov boli natrénované pomocou 58 morfo-syntaktických rysov a 124 rysov z množiny AU124. Najlepším algoritmom je SVM, hoci medzi výsledkami algoritmov neexistuje štatistiky signifikantný rozdiel. Podrobné výsledky pre všetky slovesá uvádzame v dodatku D.21 na strane 119.

Porovnaním výsledkov experimentov s množinami MU44 a AU124 konštatujeme, že modely majú takmer zhodnú úspešnosť. O málo lepší je model s MU44, ktorý okrem vyššej Accuracy na SVM má navyše užší konfidenčný interval.

8.3.2 Uporiadanie rysov podľa úspešnosti

Analogicky s optimalizovaním morfo-syntaktických rysov (popísaným v časti 8.2.2, na strane 67), sme pristúpili k optimalizovaniu sémantických rysov. Namiesto vytvárania rebríčkov A, W a G sme použili už len hladový algoritmus, ktorý skončil pridávanie rysov po prvom poklese Accuracy.

Hladový algoritmus vyberal rysy z množiny Best58 a MU44, resp. AU124. Zaujímalo nás, či sa medzi morfo-syntaktické rysy dostanú aj rysy sémantické.

Výsledky experimentu prezentuje v dodatkoch D.22 a D.23 na stranách 120 a 121. Jedná sa o zoznam rysov, ktoré hladový algoritmus vybral pre jednotlivé slovesá. Hrubo sú vyznačené sémantické rysy. Môžeme pozorovať, že sémantické rysy sa výrazne začlenili medzi rysy morfo-syntaktické.

Najlepšie rysy pre jednotlivé slovesá sme zlúčili do jedného rebríčka, podobne, ako v prípade morfo-sémantických rysov. Pre experiment s MU44 sme získali množinu 69 rysov, z ktorých bolo 18 z množiny MU44 a zvyšok z Best58. Novú množinu rysov sme nazvali **BestMU**. Pre experiment s AU124 sme získali množinu 65 rysov, z ktorých bolo 20 z množiny AU124 a zvyšok z množiny Best58. Získanú množinu rysov sme nazvali **BestAU**.

Nové množiny rysov sme použili pri trénovaní a ladení modelov klasifikátorov. Výsledky prezentujeme v tabuľkách 8.8 a 8.9.

					Zlepšenie		
Skupina	**DT**	**kNN**	**SVM**	**ADA**	**r**	**%**	**ERD**
A	86.2 ±2.6	85.5 ±2.2	87.7 ±2.6	87.1 ±2.5	7.5	10.1	36.3
B	62.1 ±4.7	62.8 ±4.0	64.4 ±4.2	63.9 ±3.6	14.5	42.4	27.8
C	65.4 ±4.4	64.6 ±4.6	68.5 ±4.7	68.4 ±4.8	24.8	75.2	42.4
Celkom	81.8 ±3.0	81.2 ±2.6	83.5 ±3.0	83.0 ±2.8	9.6	18.7	35.7

Tabuľka 8.8: Výsledky experimentu **BestMU**. Modely klasifikátorov boli natrénované pomocou 69 rysov z množiny BestMU. Najlepším algoritmom je SVM, hoci medzi výsledkami algoritmov neexistuje štatistiky signifikantný rozdiel. Podrobné výsledky pre všetky slovesá uvádzame v dodatku D.24 na strane 122.

					Zlepšenie		
Skupina	**DT**	**kNN**	**SVM**	**ADA**	**r**	**%**	**ERD**
A	86.4 ±2.5	85.8 ±1.9	87.8 ±2.4	87.2 ±2.5	7.7	10.2	37.8
B	60.9 ±4.9	60.9 ±3.6	64.1 ±4.1	63.3 ±4.4	14.2	39.2	28.1
C	65.8 ±4.4	64.8 ±4.2	68.8 ±4.3	68.2 ±4.6	25.1	75.8	43.1
Celkom	81.8 ±2.9	81.2 ±2.2	83.6 ±2.7	82.9 ±2.9	9.7	18.4	37.0

Tabuľka 8.9: Výsledky experimentu **BestAU**. Modely klasifikátorov boli natrénované pomocou 65 rysov z množiny BestAU. Najlepším algoritmom je SVM, hoci medzi výsledkami algoritmov neexistuje štatistiky signifikantný rozdiel. Podrobné výsledky pre všetky slovesá uvádzame v dodatku D.25 na strane 123.

Z prezentovaných výsledkov vyplýva, že redukované množiny rysov sú dostatnočným zdrojom informácií pre klasifikátory na to, aby dosiahli takmer úplne rovnaké úspešnosti.

Posledným experimentom, ktorý sme v tejto časti práce implementovali, bol hladový algoritmus, ktorého úlohou bolo pridávať rysy z množiny AU124 k rysom Best58. Algoritmus skončil pridávanie v momente prvého poklesu Accuracy.

Výsledkom experimentu je rebríček 19 sémantických rysov, ktoré hladový algoritmus pripojil s množine Best58. Výslednú množinu rysov sme pomenovali **GreedyAU** a použili ju na natrénovanie modelov klasifikátorov. Výsledky prezentujeme v tabuľke 8.10.

					Zlepšenie		
Skupina	**DT**	**kNN**	**SVM**	**ADA**	**r**	**%**	**ERD**
A	86.3 ±2.4	85.7 ±1.9	87.5 ±2.6	86.9 ±2.5	7.4	10.0	35.3
B	62.5 ±4.8	63.0 ±4.1	64.8 ±3.9	63.8 ±4.4	14.8	43.4	28.3
C	65.9 ±4.4	64.5 ±4.8	68.5 ±4.2	68.3 ±4.6	24.8	74.7	43.3
Celkom	81.9 ±2.8	81.4 ±2.4	83.4 ±2.8	82.7 ±2.9	9.5	18.6	35.0

Tabuľka 8.10: Výsledky experimentu **GreedyAU**. Modely klasifikátorov boli natrénované pomocou 77 rysov z množiny GreedyAU. Najlepším algoritmom je SVM, hoci medzi výsledkami algoritmov neexistuje štatistiky signifikantný rozdiel. Podrobné výsledky pre všetky slovesá uvádzame v dodatku D.26 na strane 124.

8.4 Záverečné vyhodnotenie a analýza chýb

Určenie najlepšieho modelu

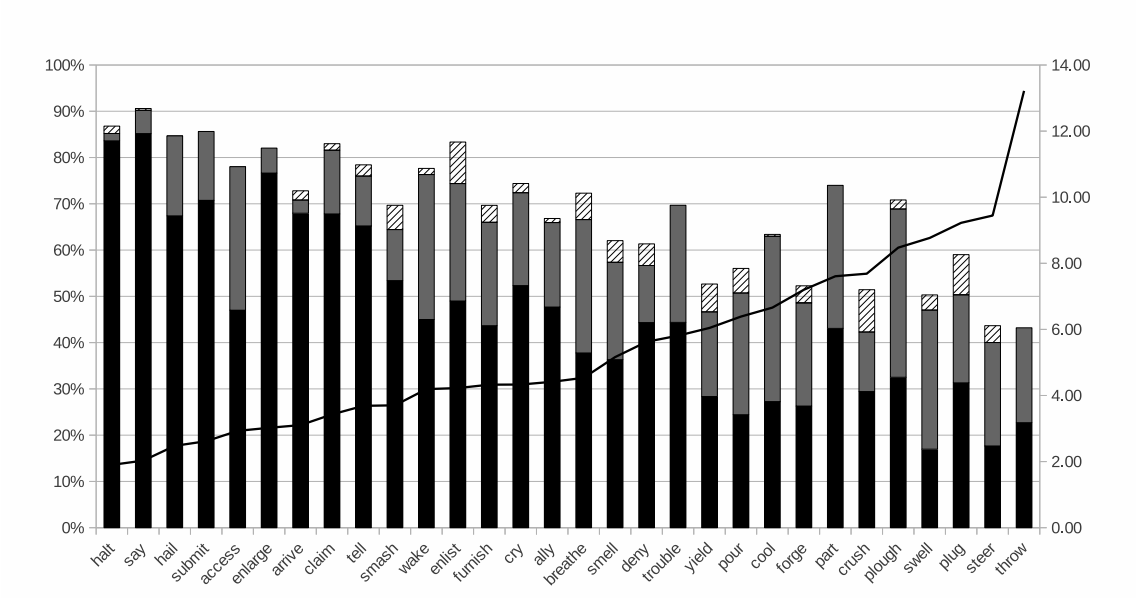
Na základe všetkých prezentovaných experimentov sme sa rozhodli za najlepšie modely prehlásiť modely klasifikátorov natrénované na množine BestAU. Výsledky experimentu BestAU sme porovnali s experimentom Best58. Výrazné (viac ako 3 percentuálne body) zvýšenie Accuracy nastalo v prípade troch slovies. Malé zlepšenie sme zaznamenali u 10 slovies. U štyroch slovies nastalo zúženie konfidenčných intervalov. Zaznamenali sme tiež 6 malých zhoršení úspešností a 9 zhoršení konfidenčných intervalov.

Dosiahnuté výsledky prezentujeme aj graficky, na obrázku 8.4. Čierna oblasť grafu vyjadruje úspešnosť baseline, šedá predstavuje zlepšenie dosiahnuté modelom Best58 a biela, vyšrafovaná oblasť predstavuje zlepšenie oproti Best58 dosiahnuté modelom BestAU. V grafe sme krivkou vyznačili i hodnotu perplexity. Môžme vidieť jasnú koreláciu medzi celkovou úspešnosťou a perplexitou.

Z dosiahnutých výsledkov vyplýva, že medzi modelmi Best58 a BestAU nemôžeme, z hľadiska štatistickej signifikantnosti, vybrať najlepší model. Dosiahnuté konfidenčné intervaly sa prekrývajú a sú navyše dosť široké.

Je dôležité si uvedomiť, že prezentované šírky konfidenčných intervalov pre skupiny A, B a C, a tiež konfidenčný interval celkového výsledku predstavujú vážené priemery konfidenčných intervalov príslušných skupín.

Zaujímalo nás, ako by vyzerali konfidenčné intervaly v prípade, ak by sme ich spočítali inou metódou. V tejto metóde sme spočítali priemernú váženú Accuracy pre každú zo skupín (A, B, C a celkovo) pre každý cyklus cross-validácie samostatne. Získali sme teda sadu hodnôt Accuracies, ktorým sme mohli pomocou t-testu spočítať konfidenčný interval. Takto získaný konfidenčný interval je užší než intervaly, ktoré sme doteraz prezentovali.



Obr. 8.4: Vylepšenia Accuracy oproti baseline pomocou modelov Best58, resp. BestAU.

V tabuľke 8.11 uvádzame porovnanie modelov Best58 a BestAU s konfidenčnými intervalmi spočítanými pomocou vyššie popísanej metódy. Na základe týchto údajov môžme prehlásiť model BestAU ako lepší (štatistiky signifikantne), než model Best58, a to pre skupinu slovies C, ktorá združuje najmenej frekventované slovesá.

V tabuľke 8.11 ďalšej uvádzame percentuálne zlepšenie modelov oproti baseline (stĺpec **Zlepšenie**) a počet percent chýb, ktoré sa podarilo odstrániť oproti baseline (stĺpec **ERD**). Tieto stĺpce sú definované analogicky ako stĺpce **ERD** a **%** vo výsledkových tabuľkách pre jednotlivé modely.

	Best58			**BestAU**		
Skupina	**Priemer**	**Zlepšenie**	**ERD**	**Priemer**	**Zlepšenie**	**ERD**
A	87.0±2.1	9.1	34.3	87.8±1.9	10.2	37.8
B	62.1±1.7	34.9	23.6	64.1±2.1	39.2	28.1
C	66.0±1.7	66.5	38.2	68.8±0.9	75.8	43.1
Celkom	82.4±1.9	16.3	33.3	83.6±1.6	18.4	37.0

Tabuľka 8.11: Porovnanie modelov Best58 a BestAU s konfidenčnými intervalmi spočítanými novým spôsobom.

Evaluácia na testovacích údajoch

Najlepšie modely (Best58 a BestAU) sme otestovali pomocou testovacích údajov, ktoré sme doteraz, počas všetkých experimentov nepoužívali. Výsledky prezentujeme v tabuľkách 8.12 a 8.13.

Skupina	Priemer	Train	Test	MULTI
A	87.0 ±2.4	91.0	83.5	83.7
B	62.1 ±3.8	69.6	56.4	60.0
C	66.0 ±4.8	80.4	62.0	67.3
Celkom	82.4 ±2.8	87.6	78.6	79.6

Tabuľka 8.12: Výsledky modelu **Best58** na testovacích údajoch. Modely klasifikátorov boli natrénované pomocou 58 rysov z množiny Best58, pomocou algoritmu SVM. Podrobné výsledky pre všetky slovesá uvádzame v dodatku D.27 na strane 125.

Skupina	Priemer	Train	Test	MULTI
A	87.8 ±2.4	91.8	86.4	86.8
B	64.1 ±4.1	74.3	58.8	62.8
C	68.8 ±4.3	82.1	66.9	71.8
Celkom	83.6 ±2.7	89.0	81.6	82.8

Tabuľka 8.13: Výsledky modelu **BestAU** na testovacích údajoch. Modely klasifikátorov boli natrénované pomocou 58 rysov z množiny BestAU, pomocou algoritmu SVM. Podrobné výsledky pre všetky slovesá uvádzame v dodatku D.28 na strane 126.

V stĺpci **Priemer** uvádzame priemernú Accuracy spolu s konfidenčným intervalom tak, ako bola uvedená v predchádzajúcich experimentoch. Všetky ostatné stĺpce sa týkajú evaluácie modelu na testovacích údajoch. Pri trénovaní modelov, ktoré sa testovali na trénovacích údajoch, sme už nepoužívali cross-validáciu, ale použili sme všetky dostupné trénovacie údaje na natrénovanie klasifikátorov.

V stĺpci **Train** uvádzame Accuracy, ktorú klasifikátor dosiahol na trénovacích údajoch. Táto hodnota nie je veľmi zaujímavá, pre úplnosť ju ale uvádzame. Vyjadruje koľko percent z trénovacích údajov klasifikátor klasifikoval správne, pričom bol klasifikovaný na rovnakých údajoch.

V stĺpci **Test** uvádzame najdôležitejšiu hodnotu – úspešnosť klasifikátora na testovacích inštanciách. Tieto inštancie klasifikátor *nevidel* počas trénovania, ani počas ladenia parametrov.

Okrem štandardnej miery Accuracy sme využili to, že kolekcia VPS obsahuje údaje týkajúce sa testovacích inštancií, ktoré obsahujú aj zoznam prípustných patternov, ktoré skúsený lexikograf označil za možné patterny pre konkrétnu inštanciu. (Popis uvádzame v sekcii 5.4.) Lexikograf pripustil, že určitá inštancia mohla byť anotovaná viac než jedným patternom. V stĺpci **MULTI** sme spočítali hodnotu Accuracy, pre ktorú sa pri výpočte za správny výsledok považuje, ak klasifikátor klasifikuje inštanciu patternom vyskytujúcim sa v množine prípustných patternov.

V tabuľkách D.27 a D.28 sme okrem vyššie popísaných stĺpcov pripojili aj podrobnú informáciu o parametroch algoritmu SVM, ktorý dosiahol najlepšie úspešnosti na trénovacích údajoch.

Na záver tejto časti prezentujeme, aké zlepšenie na jednotlivých skupinách sme dosiahli pomocou sémantických rysov, tj. aké zlepšenie dosiahol model BestAU oproti modelu Best58. Výsledky porovnania prezentujeme v tabuľke 8.4

Analýza chýb

Stručná analýza chýb je popísaná v článku [56].

Skupina	Train	Test	MULTI
A	1.0	3.5	3.8
B	3.4	4.2	4.6
C	4.4	7.9	6.8
Celkom	1.4	3.8	4.0

Tabuľka 8.14: Percentuálne zlepšenie dosiahnuté použitím sémantických rysov (model BestAU) oproti modelu používajúcemu len morfo-syntaktické rysy (Best58).

Modely klasifikátorov, ktoré sme v práci trénovali a testovali nepracovali s definíciami patternov. Toto rozhodnutie sme urobili so zámerom zistiť, ako dobre sú navrhnuté množiny rysov použité pri trénovaní. Tento prístup spolu so značnou riedkosťou údajov mali za následok niekoľko systematických chýb.

Najväčšia chyba, ktorej sa klasifikátory dopúšťali, bolo zlé klasifikovanie patternov, ktoré popisujú participiálne formy slovesa. Ak tieto patterny neboli dostatočne zastúpené v trénovacích inštanciách, klasifikátory sa nenaučili najdôležitejšiu vedomosť potrebnú pre správnu klasifikáciu týchto patternov, a síce, že sloveso je v participiu a zároveň nie je použité v pasívnej vete, ale namiesto toho je už použité ako adjektívum alebo podstatné meno (napríklad *cooling* v *cooling towers*). Klasifikátor často priradzoval patterny definované pre participiá inštanciám, v ktorých slovesá vôbec neboli vo forme participií.

Ďalší nedostatok klasifikátorov spočíval v chýbajúcom rozpoznávaní frazémov. Vďaka veľkej riedkosti údajov ostali idiómy často nerozpoznané alebo boli interpretované doslova. Napríklad inštancia

> *This organisation happily **ploughs** a furrow totally at odds with the notion of free trade.*

bola označená patternom vyjadrujúcim poľnohospodársku činnosť. Tento problém sa netýka len idiómov, pretože mnoho patternov s obmedzenou schopnosťou tvorenia kolokácií má definované lexikálne množiny. Podstatné mená v týchto množinách sú často veľmi heterogénne, zahrňujúce niekoľko sémantických typov a preto je ich asociácia so sémantickými typmi irelevantná.

Nie je prekvapením, že niektoré chyby klasifikátora boli spôsobené redukciou patternov. Napriek tomu prekvapivo často klasifikoval inštancie pomocou všeobecnejších patternov, aj keď existoval vhodnejší a viac špecifickejší pattern. Táto skutočnosť sa neodrazila ani v meraní úspešnosti implementovanej pomocou množiny prípustných patternov, pretože všeobecnejšie patterny boli príliš všeobecné na to, aby ich lexikograf zaradil k prípustným patternom.

Zaujímavé pozorovanie sa týka aj sémantickej modulácie. Typickým príkladom je inštancia slovesa *halt*:

> *And even Crown Prince Rupprecht, far removed from Verdun, had warned him days before the offensive began that the advance would be **halted** by flanking fire from the Left Bank.*

Patterny slovesa *halt* rozlišujú medzi abstraktnými procesmi (napríklad *finančná kríza*) a medzi skupinami ľudí v militantnom kontexte a dopravnými prostriedkami, ktoré môžu byť zastavené. V uvedenej inštancii je slovo *advance* proces, ale v skutočnosti sa hovorí o mužoch a dopravných prostriedkoch, ktorí *postupujú*. Sémantická modulácia je zdrojom častých chýb anotátorov. Klasifikátory tento druh chýb robili tiež pomerne často.

Záver

Cieľom tejto diplomovej práce bolo navrhnúť, implementovať a empiricky evaluovať klasifikátory pre rozpoznávanie sémantických patternov anglických slovies. Ako trénovacie a testovacie údaje sme používali konkordancie z kolekcie VPS.

Úlohu sme riešili pomocou metód strojového učenia s učiteľom, kontrétne sme experimentovali s algoritmom rozhodovacích stromov, algoritmom k najbližších susedov (kNN), algoritmom podporných vektorov (SVM) a algoritmom Adaboost (pomocou rozhodovacích stromov).

V práci sme sa, okrem iného, zamerali na návrh vhodnej množiny rysov pre strojové učenie (*feature selection*). Navrhli a definovali sme množinu 149 morfo-syntaktických rysov a tri množiny sémantických rysov. Vykonali sme množstvo experimentov v snahe optimalizovať navrhnuté množiny rysov tak, aby modely klasifikátorov dosahovali čo najlepšie úspešnosti.

Výsledkom našich experimentov je prehlásenie dvoch množín rysov (čiste morfo-syntaktických, resp. morfo-syntaktických a sémantických) za množiny najvhodnejšie k trénovaniu klasifikátorov. Pomocou nich sme dosiahli výrazné, štatisticky signifikantné, zlepšenie oproti baseline. Myslíme si, že nedostatok vhodných trénovacích inštancií tvorí hlavný limit vo výkonosti našich klasifikátorov.

Naším cieľom bolo tiež overiť, či sémantické prototypy použité priamo ako rysy budú užitočné pre algoritmy strojového učenia. Naše výsledky sú zhodné so závermi skôr publikovaných štúdií (napríklad [57]), ktoré potvrdzujú fakt, že morfo-syntaktické rysy sú najdôležitejšie pre sémantickú desambiguáciu pomocou strojového učenia s učiteľom. Napriek tomu, pre *niektoré* slovesá hrajú sémantické rysy dôležitú úlohu.

Hlavné výsledky tejto práce boli tiež zhrnuté v článku [56].

Zoznam použitej literatúry

[1] *Word Sense Disambiguation*, Webová enclyklopédia Wikipedia. `http://en.wikipedia.org/wiki/Word_sense_disambiguation`

[2] Agirre, E., Edmonds P. *Word Sense Disambiguation: Algorithms and Applications*, Springer, 2007.

[3] Weaver, W. *Translation.* In Locke, W. N.; Booth, A. D. *Machine Translation of Languages: Fourteen Essays.* MIT Press, Cambridge, MA, 1949.

[4] Cinková S., Holub M., Kríž V. *Managing Uncertainty in Semantic Tagging.* To appear. Accepted for EACL 2012, Avignon, France.

[5] Cinková S., Kríž V., Holub, M. *Optimizing semantic granularity for NLP - report on a lexicographic experiment.* To appear. Accepted for the 15th Euralex International Congress, Oslo, Norway, 2012.

[6] Hanks, P. forthcoming. *Lexical Analysis: Norms and Exploitations.* MIT Press.

[7] Hanks, P., Pustejovsky J. *A pattern dictionary for natural language processing.* Revue Francaise de linguistique applique, 10(2), 2005.

[8] Webové stránky medzinárodného projektu CPA. `http://nlp.fi.muni.cz/projekty/cpa`

[9] Cinková, S., Hanks, P.: *Validation of Corpus Pattern Analysis - Assigning pattern numbers to random verb samples.* Annotation manual. 2010. `http://nlp.fi.muni.cz/projekty/cpa/`

[10] Ruppenhofer J., Ellsworth M., Petruck, M. R. L., Johnson, C. R., Scheffczyk, J. *FrameNet II: Extended Theory and Practice.* ICSI, University of Berkeley, September 2010.

[11] Fellbaum C. *WordNet. An Electronic Lexical Database.* MIT Press, Cambridge, MA, 1998.

[12] Palmer M., Gildea D., Kingsbury P. *The proposition bank: A corpus annotated with semantic roles.* Computational Linguistics Journal, 31(1). 2005.

[13] Weischedel R., Palmer M., Marcus M., Hovy E., Pradhan S., Ramshaw L., Xue N., Taylor A., Kaufman J., Franchini M., El-Bachouti M., Belvin R., Houston A. *OntoNotes release 4.0*, Linguistic Data Consortium, 2011.

[14] Smejkalová L. *Typické vzory užívaní anglických sloves*, Diplomová práca, MFF UK, Praha, 2010.

[15] Cinková, S., Holub, M., Rychlý, P., Smejkalová, L., Šindlerová, J. *Can Corpus Pattern Analysis Be Used in NLP?* In Sojka, P., Horák, A., Kopeček, I., Pala, K.: *Text, Speech and Dialogue*. Proceedings of the 13th International Conference, TSD 2010, Brno, Czech Republic. Springer, Berlin/Heidelberg, 2010.

[16] Jurafsky D., Martin, J. H. *Speech and language processing. An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition.* Prentice Hall, 2009.

[17] Sinclair J.: *The lexical item.* In Hanks, P. (ed.), *Lexicology: Critical Concepts in Linguistics.* 6 volumes. Routledge, 2008.

[18] Webové stránky Britského národného korpusu. `http://www.natcorp.ox.ac.uk/corpus/`

[19] Cinková S., Holub M., Rambousek A., Smejkalová L. *A database of semanatic clusters of verb usages.* To appear. Accepted for LREC 2012, Istanbul, Turkey.

[20] Carletta, J. *Assessing agreement on classification tasks: The kappa statistic.* Computational Linguistics, 22(2), s. 249–254. 1996.

[21] Fleiss, J. L. *Measuring nominal scale agreement among many raters.* Psychological Bulletin 76(5), s. 378–382. 1971.

[22] J. R. Quinlan: *Induction of decision trees.*. Machine Learning, s. 81–106. 1986.

[23] J. R. Quinlan: *C4.5: Programs for Machine Learning*, Morgan Kaufman. 1993.

[24] J. R. Quinlan: *Data mining tools see5 and c5.0.* `http://www.rulequest.com/see5-info.html`

[25] Bremner, D., Demaine, E., Erickson, J., Iacono, J., Langerman, S., Morin, P., Toussaint, G. *Output-sensitive algorithms for computing nearest-neighbor decision boundaries.* Discrete and Computational Geometry 33(4), s.593–604. 2005.

[26] Deza, E., Deza M. M. *Encyclopedia of Distances.* Springer, 2009.

[27] Boser, B. E., Guyon, I., Vapnik, V. *A training algorithm for optimal margin classifiers.* In *Computational Learing Theory*, s. 144–152. 1992.

[28] Cortes, C., Vapnik, V. *Support-vector networks.* In *Machine Learning*, 20(3), s. 273–297. 1995.

[29] Vapnik, V. *Statistical Learning Theory.* Wiley-Interscience, 1989.

[30] Yoav Freund, Robert E. Schapire: *A Decision-Theoretic Generalization of on-Line Learning and an Application to Boosting*, In *Proceedings of the Second European Conference on Computational Learning Theory.* Springer-Verlag London, UK. 1995.

[31] Lenat, D., Guha, R. V. *Building Large Knowledge-Based Systems*, Addison-Wesley, 1989.

[32] Lesk, M. *Automatic sense disambiguation using machine readable dictionaries: How to tell a pine cone from an ice cream cone.* In *Proc. of SIGDOC-86: 5th International Conference on Systems Documentation*, Toronto, Canada, 1986.

[33] Ponzetto, S. P., Navigli R. *Knowledge-rich Word Sense Disambiguation rivaling supervised systems.* In *Proc. of the 48th Annual Meeting of the Association for Computational Linguistics* (ACL), 2010.

[34] Yarowsky, D. *Unsupervised word sense disambiguation rivaling supervised methods.* In *Proc. of the 33rd Annual Meeting of the Association for Computational Linguistics.* 1995.

[35] Schütze, H. *Automatic word sense discrimination.* Computational Linguistics, 24(1), s. 97–123. 1998.

[36] Navigli, R. *Word Sense Disambiguation: a Survey.* ACM Computing Surveys, 41(2), s. 1-69. ACM Press, 2009,

[37] Semecký J. *Verb Valency Frames Disambiguation*, Dizertačná práca, MFF UK, 2007.

[38] Panevová J. *On verbal frames in Functional generative description* In *Prague Bulletin of Mathematical Linguistics*, č. 22, s. 3–40, Praha, 1974.

[39] Panevová J. *Formy a funkce ve stavbě české věty.* Academia, Praha, 1980.

[40] Panevová J. *Valency frames and the meaning of the sentence.* In *The Prague School of Structural and Functional Linguistics*, s. 223–243, Praha, 1994.

[41] Webové stránky Stanfordského parsera.
`http://nlp.stanford.edu/software/lex-parser.shtml`

[42] Webové stránky Charniak-Johnosonovho parsera.
`http://www.cog.brown.edu/ mj/Software.htm`

[43] Webové stránky McDonaldovho parsera.
`http://sourceforge.net/projects/mstparser/`

[44] Webové stránky projektu TectoMT.
`http://ufal.mff.cuni.cz/tectomt/`

[45] Webové stránky projektu Sketch Engine.
`http://www.sketchengine.co.uk/`

[46] Webové stránky projektu Stanford NE Recognize.
`http://nlp.stanford.edu/software/CRF-NER.shtml`

[47] Webové stránky korpusu Penn Tree Bank.
`http://www.cis.upenn.edu/ treebank/`

[48] Holub M. *Term project specification: Semantic Pattern Classification.* Výukový materiál. 2011.
`http://ufal.mff.cuni.cz/hladka/info-on-lecture/ML_project.html`

[49] P. Vossen, L. Bloksma, H. Rodriquez, S. Climent, N. Calzolari, A. Roventini, F. Bertagna, A. Alonge, W. Peters: *The EuroWordNet Base Concepts and Top Ontology.* EuroWordNet (LE-4003) Deliverable D017D034D036, University of Amsterdam, 1998.
`http://www.vossen.info/docs/1998/D017.pdf`

[50] E. Bick: *Automatic parsing of Portuguese*. Publikované v Garcia, Laura Sanchez (ed.), /Anais / II Encontro a para o Processamento Computacional de Portugułs Escrito e Falado/. Curitiba: CEFET-PR, 1996.

[51] E. Bick: *Semantic prototype tags for nouns*. Dokumentácia k sémantickému lexikónu, 2009.
`http://beta.visl.sdu.dk/semantic prototypes overview.pdf`

[52] Webové stránky štatistikého systému R.
`http://www.r-project.org/`

[53] T. M. Therneau and B. Atkinson: *Package 'rpart'*. Reference manual. 2012.
`http://cran.r-project.org/web/packages/rpart/rpart.pdf`

[54] E. Dimitriadou, K. Hornik, F. Leisch, D. Meyer and A. Weingessel: *Package 'e1071'*. Reference manual. 2011.
`http://cran.r-project.org/web/packages/e1071/e1071.pdf`

[55] E. Alfaro-Cortes, M. Gamez-Martinez and N. Garcia-Rubio: *Package 'adabag'*. Reference manual. 2011.
`http://cran.r-project.org/web/packages/adabag/adabag.pdf`

[56] Holub, M., Kríž, V., Cinková, S., Bick, E. *Automatic Classification of Verb Semantic Patterns*. Submitted to the First Joint Conference on Lexical and Computational Semantics (*SEM), 2012.

[57] Marquez, L., Escudero, G., Martinez, D., Rigau, D. *Supervised Corpus-Based Methods for WSD*. In Agirre, E., Edmonds P. (editors) *Word Sense Disambiguation: Algorithms and Applications*, Springer, 2007.

Zoznam tabuliek

Dodatok A

Zoznam morfologických značiek

Morfologické značky z korpusu Penn Tree Bank [47] boli použité ako zdroj morfologických informácií o každom slove. V tomto dodatku uvádzame zoznam značiek, s ktorými pracovala morfologická analýza.

Zn.	**Význam**	**Zn.**	**Význam**
CC	Priraďovacia spojka	RB	Adverbium
CD	Radová číslovka	RBR	Adverbium, komparatív
DT	Determinátor	RBS	Adverbium, superlatív
EX	Existenciálne *there*	RP	Častica
FW	Cudzie slovo	SYM	Symbol
IN	Predložka alebo	TO	infinitívna častica *to*
	podraďovacia spojka	UH	Spojka
JJ	Adjektívum	VB	Sloveso, základný tvar
JJR	Adjektívum, komparatív	VBD	Sloveso, minulý čas
JJS	Adjektívum, superlatív	VBG	Sloveso, gerundium alebo
LS	Položka zoznamu		príčastie prítomné
MD	Modálne sloveso	VBN	Sloveso, príčastie minulé
NN	Substantívum, singulár	VBP	Sloveso, nie v 3. osobe
NNS	Substantívum, plurál		singuláru prítomného času
NP	Vlastné substantívum, singulár	VBZ	Sloveso, v 3. osobe
NPS	Vlastné substantívum, plurál		singuláru prítomného času
PDT	Predeterminátor	WDT	Vzťažné zámeno
POS	Privlastňovacia koncovka	WP	Opytovacie zámeno
PP	Osobné zámeno	WP$	Privlastňovacie zámeno
PP$	Privlastňovacie zámeno	WRB	Opytovacia príslovka

Tabuľka A.1: Zoznam značiek, s ktorými pracuje morfologická analýza

Dodatok B

Stanfordské závislosti

Stanfordské závislosti je označenie výstupu Stanfordského parsera [41]. Hoci sa pôvodne jednalo o zložkový parser, bol doplnený o možnosť prevodu zložkových stromov do závislostného formalizmu, ktorý umožňuje zachytiť viac sémantickej informácie pre danú vetu.

V súčasnosti existuje niekoľko aplikácií, ktoré umožňujú previesť výstupy rôznych parserov do podoby stanfordských závislostí. Tým sa tento formát stáva veľmi vhodným na ďalšie strojové spracovanie, pretože umožňuje použiť rôzne parsery a ich výstup (po prevedení na stanfordské závislosti) ďalej spracovávať jedným spôsobom.

Výstupom parsera je pre danú vetu zoznam závislostí v tvare

`DruhZávislosti` (*Nadradený token*-Poradie, *Podradený token*-Poradie)

Takýto zápis vyjadruje informáciu, že v závislostnom strome, ktorý môžeme považovať za orientovaný graf, existuje hrana vedúca z nadradeného tokenu do podradeného tokenu. Všetky tokeny sú zároveň očíslované podľa ich pozície vo vetnom slovoslede, aby nedošlo k zámene s iným tokenom rovnakého tvaru.

Existujú štyri typy stanfordských závislostí, ktoré sa od seba líšia v niekoľkých detailoch:

Základný typ. Zoznam závislostí vytvára stromovú štruktúru a obsahuje len základné závislosti. Každé slovo vo vete sa objaví v strome ako samostatný token. To nemusí byť vždy praktické, napríklad podstatné meno s predložkou nie je priamo závislé na slovese, ale medzi nimi je ešte vložená daná predložka.

Redukované závislosti. Stromová štruktúra tu nie je dôsledne zachovaná a môžu sa dokonca vyskytovať aj cykly. Predložkové skupiny popísané v predchádzajúcom type sú v tomto prípade kódované praktickejším spôsobom, ktorý čitateľ ľahko pochopí z nasledujúceho príkladu:

Základný typ	Redukované závislosti
`prep`(links, with), `pobj`(with, Bush)	`prep_with`(links, Bush)

Redukované závislosti s propagáciou koordinácií. Okrem vlastností popísaných pre druhý typ závislostí umožňuje tento typ ešte aj *roznásobovanie* koordinácií. Napríklad, ak existuje hrana (x, y) a koordinácia (y, z), do zoznamu závislostí sa doplní aj hrana (x, z), čo je veľmi praktické pri strojovom spracovaní závislostí.

Redukované závislosti so zachovanou stromovou štruktúrou. Zo zoznamu závislostí sa odstránia všetky hrany, ktoré narušujú stromovú štruktúru. Hrany s roznásobenou koordináciou popisované v treťom type by sa teda v tomto type neobjavili.

Medzi prednosti Stanfordské parsera patrí tiež jeho jednoduchá obsluha a takmer žiadna potreba predspracovania vstupu. Parser totiž obsahuje vlastný tokenizátor i morofologickú analýzu.

V ďalšom texte uvádzame zoznam závislostí, ktoré môžu byť výstupom Stanfordského parsera:

dep - dependent

1. **aux** - auxiliary
   (a) **auxpass** - passive auxiliary
   (b) **cop** - copula
2. **conj** - conjunct
3. **cc** - coordination
4. **arg** - argument
   (a) **subj** - subject
      i. **nsubj** - nominal subject
         A. **nsubjpass** - passive nominal subject
      ii. **csubj** - clausal subject
   (b) **comp** - complement
      i. **obj** - object
         A. **dobj** - direct object
         B. **iobj** - indirect object
         C. **pobj** - object of preposition
      ii. **attr** - attributive
      iii. **ccomp** - clausal complement with internal subject
      iv. **xcomp** - clausal complement with external subject
      v. **compl** - complementizer
      vi. **mark** - marker (word introducing an advcl)
      vii. **rel** - relative (word introducing a rcmod)
      viii. **acomp** - adjectival complement
   (c) **agent** - agent
5. **ref** - referent
6. **expl** - expletive (expletive there)
7. **mod** - modifier
   (a) **advcl** - adverbial clause modifier
   (b) **purpcl** - purpose clause modifier
   (c) **tmod** - temporal modifier
   (d) **rcmod** - relative clause modifier
   (e) **amod** - adjectival modifier
   (f) **infmod** - infinitival modifier
   (g) **partmod** - participial modifier
   (h) **num** - numeric modifier
   (i) **number** - element of compound number
   (j) **appos** - appositional modifier

(k) **nn** - noun compound modifier

(l) **abbrev** - abbreviation modifier

(m) **advmod** - adverbial modifier

i. **neg** - negation modifier

(n) **poss** - possession modifier

(o) **possessive** - possessive modifier ('s)

(p) **prt** - phrasal verb particle

(q) **det** - determiner

(r) **prep** - prepositional modifier (also pmod, sometimes)

8. **sdep** - semantic dependent

(a) **xsubj** - controlling subject

Dodatok C

Lexikón sémantických prototypov

V nasledujúcom texte popisujeme sémantické značky použité v Lexikóne sémantických prototypov (LSP) [50, 51]. V prvej časti uvádzame zoznam všetkých prototypov a zastrešujúce prototypy, ktoré môžeme chápať ako všeobecnejšie koncepty, vďaka ktorým môžeme výrazne znížiť počet sémantických značiek, ak nepotrebujeme rozoznávať významy slov do tak podrobnej miery.

V druhej časti definujeme systém značiek, ktoré sa používajú na anotovanie vlastných podstatných mien v lexikóne. Z historických dôvodov sú postavené mimo hlavnú štruktúru prototypov. Do istej miery ich ale môžeme zlúčiť.

V poslednej časti definujeme hlavné zastrešujúce prototypy, ktoré zjednocujú vybrané zastrešujúce prototypy a značky pre vlastné podstatné mená do najvyššej úrovne v hierarchii prototypov a tvora množinu 22 hlavných zastrešujúcich prototypov.

C.1 Prototypy

LSP obsahuje 179 sémantických prototypov. Pre každý prototyp je definovaný všeobecnejší zastrešujúci prototyp, ktorý zjednocuje vybrané prototypy do niekoľkých hlavných skupín.

V nasledujúcom prehľade uvádzame zoznam prototypov spolu s ich zaradením do zastrešujúcich prototypov. Vždy platí konvencia v pomenovaní prototypov, že názov zastrešujúceho prototypu je zároveň prefixom všetkých prototypov v danej skupine.

Zvieracie prototypy

Umbrella	Prototyp	Názov
A	–	Zoológia
A	AA	Skupina zvierat
A	Adom	Domestifikované zvieratá
A	Aich	Vodné zvieratá
A	Amyth	Mytologické zvieratá
A	Azo	Divoké zvieratá
A	Aorn	Vtáky
A	Aent	Hmyz
A	Acell	Bunky živých organizmov

Rastlinné prototypy

Umbrella	Prototyp	Názov
B	–	Rastlina
B	BB	Skupina rastlín
B	Btree	Stromy
B	Bflo	Kvety
B	Bbush	Kroviny

Ľudské prototypy

Umbrella	Prototyp	Názov
H	–	Ľudská bytosť
H	HH	Skupina ľudí
H	Hbio	Biologické slová
H	Hfam	Rodina
H	Hideo	Ideológie (*hinduista*)
H	Hmyth	Mytologické postavy
H	Hnat	Národnosti
H	Hprof	Profesie, športovci
H	Hsick	Chorí ľudia (*diabetik*)
H	Htit	Tituly

Miesta a priestory

Umbrella	Prototyp	Názov
L	–	Miesta
L	La	Zvieracie miesta (*nora*)
L	Labs	Abstraktné miesta
L	Lbar	Bariéry
L	Lciv	Mestá, okresy, krajiny
L	Lcover	Úkryty
L	Lh	Ľudské miesta (*obývačka*)
L	Lopening	Otvorené priestory
L	Lpath	Cesty, ulice
L	Lstar	Kozmické objekty
L	Lsurf	Povrchy
L	Ltip	Vrcholy
L	Ltop	Prírodné miesta
L	Ltrap	Pasce
L	Lwater	Rieky, jazerá, moria

Dopravné prostriedky

Umbrella	Prototyp	Názov
V	–	Dopravné prostriedky
V	VV	Skupina dopr. prostriedkov
V	Vground	Pozemné
V	Vwater	Vodné
V	Vair	Letecké

Abstraktné prototypy

Umbrella	Prototyp	Názov
ac	–	Abstraktné počítateľné slová
ac	ac-cat	Kategoriálne slová (*metafora*)
ac	ac-sign	Znaky, symboly
am	–	Abstraktné nepočítateľné slová

Akcie

Umbrella	Prototyp	Názov
act	–	Akcie
act	act-d	Robenie
act	act-move	Pohyb
act	act-s	Komunikácia (*rozhovor*)
activity	–	Aktivita

Anatomické prototypy

Umbrella	Prototyp	Názov
an	–	Anatomický termín
an	anmov	Končatiny
an	anorg	Orgány
an	anost	Kosti
an	anzo	Anatómia zvierat
an	anorn	Anatómia vtákov
an	anich	Anatómia rýb
an	anent	Anatómia hmyzu
an	anbo	Anatómia rastlín

Veci

Umbrella	Prototyp	Názov
cc	–	Konkrétne počítateľné
cc	cc-a	Artefakty
cc	cc-cord	Šnúry, laná
cc	cc-fire	Oheň
cc	cc-handle	Rukoväť
cc	cc-light	Svetlo
cc	cc-org	Organické veci
cc	cc-particle	Elementárne častice
cc	cc-r	Veci na čítanie
cc	cc-rag	Veci z textilu (*koberec*)
cc	cc-stick	Palica
cc	cc-stone	Kamene

Substancie

Umbrella	Prototyp	Názov
cm	–	Konkrétne nepočítateľné substancie
cm	cm-h	Vyrobené človekom (*betón*)
cm	cm-chem	Chemické zlúčeniny
cm	cm-gas	Plyny
cm	cm-liq	Kvapaliny
cm	cm-rem	Lieky, hygienické veci

Oblečenie

Umbrella	Prototyp	Názov
cloH	–	Oblečenie
cloH	cloH-beauty	Šperky
cloH	cloH-hat	Klobúky
cloH	cloH-shoe	Topánky

Kolektívne prototypy

Umbrella	Prototyp	Názov
coll	–	Kolektív, skupina
coll	coll-cc	Kolektívne veci
coll	coll-sem	Kolekcie
coll	coll-tool	Skupiny nástrojov

Ďalšie prototypy

Umbrella	Prototyp	Názov
amount	–	Kvantity
build	–	Stavby
col	–	Farby
con	–	Kontajnery
conv	–	Pravidlá, zákony
cur	–	Meny
dance	–	Tance
dir	–	Smery
domain	–	Domény
drink	–	Nápoje
fight	–	Konflikty
fruit	–	Ovocie
furn	–	Nábytok
geom	–	Geometrické tvary
inst	–	Inštitúcie
ism	–	Ideológie
ling	–	Jazyky
mach	–	Stroje
mat	–	Materiály
mat	mat-cloth	Textílie
meta	–	Meta substantíva (*umenie*)
mon	–	Finančné čiastky
month	–	Mesiace v roku
occ	–	Príležitosti
per	–	Periodické udalosti
pict	–	Obrazy
pos	–	Polohy
pos	pos-a	Anatomické polohy
pos	pos-soc	Sociálne polohy
process	–	Procesy (*stagnácia*)
sick	–	Choroby
spice	–	Korenia
sport	–	Športy
talk	–	Diskusie
temp	–	Body v čase
therapy	–	Terapie
tube	–	Tunely, rúry
unit	–	Jednotky

Časové prototypy

Umbrella	Prototyp	Názov
dur	–	Trvanie (*okamih*)
event	–	Udalosť

Vlastnosti

Umbrella	Prototyp	Názov
f	–	Vlastnosti
f	f-an	Vlastnosti anatomické
f	f-c	Všeobecné počítateľné vlastnosti
f	f-h	Ľudské vlastnosti (nie psych.)
f	f-q	Vlastnosti kvantitatívne
f	f-phys	Vlastnosti fyzické
f	f-right	Vlastnosti sociálne (*spravodlivosť*)
f	f-surf	Vlastnosti povrchov

Jedlo

Umbrella	Prototyp	Názov
food	–	Jedlo
food	food-c	Počítateľné jedlá (*nanuk*)
food	food-h	Uvarené jedlá

Koncepty

Umbrella	Prototyp	Názov
game	–	Hry
genre	–	Umelecké žánre

Časti

Umbrella	Prototyp	Názov
part	–	Časti
part	part-build	Časti budovy
part	part-V	Časti dopravných prost.
piece	–	Malá časť niečoho

Vnímanie

Umbrella	Prototyp	Názov
percep	–	Vnímanie
percep	percep-f	Pocity
percep	percep-l	Zvuky
percep	percep-o	Zápachy, vône
percep	percep-t	Chute
percep	percep-w	Očné vnemy

Sémantické produkty

Umbrella	Prototyp	Názov
sem	–	Výtvory mysle
sem	sem-c	Koncepty, plány, systémy
sem	sem-l	Hudba
sem	sem-nons	Nonsens
sem	sem-r	Texty (*román*)
sem	sem-s	Reč (*prejav*)
sem	sem-w	Vizuálne diela (*film*)

Okolnosti dejov

Umbrella	Prototyp	Názov
sit	–	Situácia
state	–	Stav niečoho
state	state-h	Ľudský stav

Nástroje

Umbrella	Prototyp	Názov
tool	–	Nástroje
tool	tool-cut	Krájanie, rezanie
tool	tool-shoot	Streľné zbrane
tool	tool-mus	Hudobné nástroje
tool	tool-sail	Nástroje pre plavbu

Počasie

Umbrella	Prototyp	Názov
wea	–	Počacie
wea	wea-c	Počítateľné fenomény
wea	wea-rain	Zrážky
wea	wea-wind	Vetry, búrky

C.2 Značky pre rozpoznávač menných entít

LSP obsahuje niekoľko tisíc vlastných podstatných mien, ktoré sú anotované niektorou z nasledujúcich značiek. Značky pre rozpoznávač menných entít tvoria, podobne ako prototypy a zastrešujúce prototypy, hierarchickú štruktúru. Niektoré značky majú prefix zhodný zo zastrešujúcimi prototypmi a môžeme ich preto bez problémov zaradiť do systému prototypov.

- **hum**. Mená osôb. Obsahuje tieto podkategórie:
  - **fem**. Ženské mená.
  - **masc**. Mužské mená.
  - **Hfunc**. Historické mená funkcií (*Rudolf II.*).
- **top**. Vlastné mená geografických miest. Obsahuje tieto podkategórie:
  - **Lcountry**. Územné jednotky.
  - **Lmountain**. Pohoria.
  - **Lregion**. Regióny.
  - **Lriver**. Rieky.
  - **Ltown**. Mestá.
- **inst**. Názvy inštitúcií – ministerstvá, školy, obchody.
- **org**. Názvy organizácií – spoločnosti, kluby, asociácie. Obsahuje tieto podkategórie:
  - **media**. Mediálne organizácie.
  - **party**. politické strany.

- **brand**. Názvy značiek áut, oblečenia, počítačov, ...
- **tit**. Názvy umeleckých diel. Obsahuje jednu podkategóriu:
  - **prize**. Názvy ocenení.

C.3 Hlavné zastrešujúce prototypy

V LSP je definovaných 22 *hlavných zastrešujúcich prototypov*, ktoré umožňujú anotovať väčšinu lexikónu pomocou 22 skupín prototypov. V definíciách hlavných zastrešujúcich prototypov sa okrem zastrešujúcich prototypov vyskytujú aj niektoré značky pre rozpoznávač menných entít. Sú definované nasledujúcim spôsobom:

1. **A** = A
2. **B** = B
3. **H** = H + org + inst + hum + fem + masc
4. **L** = L + build + dir
5. **V** = V
6. **ac** = ac + am + geom + meta
7. **act** = act + event + process + occ + talk + fight
8. **an** = an
9. **cc** = cc + con + furn
10. **clo** = clo
11. **cm** = cm + mat
12. **coll** = coll + part + piece
13. **domain** = domain + ism + genre + ling + sport + game
14. **f** = f + col
15. **food** = food + drink + fruit
16. **percep** = percep
17. **sem** = sem + conv
18. **sit** = sit + state + pos + system
19. **temp** = temp + per + month
20. **tool** = tool + mach
21. **unit** = unit + cur + dur + amount + mon
22. **wea** = wea

Dodatok D

Podrobné výsledky

D.1 Základné charakteristiky 30 cieľových slovies

	Počet konkordancií			Frekvencia v korpuse			Skupiny	
Sloveso	**Ref. vzorka**	**Multi. vzorka**	**Počet celkom**	**Počet konk.**	**Podiel konk.**	**Váha slovesa**	**Skupina**	**Váha v skupine**
access	300	50	350	455	0.010%	0.29%	C	4.04%
ally	250	50	300	386	0.008%	0.24%	C	3.42%
arrive	250	50	300	6110	0.131%	3.83%	B	31.22%
breathe	350	50	400	950	0.020%	0.60%	C	8.43%
claim	500	50	550	12517	0.268%	7.85%	A	9.73%
cool	300	50	350	575	0.012%	0.36%	C	5.10%
crush	350	50	400	435	0.009%	0.27%	C	3.86%
cry	250	50	300	1200	0.026%	0.75%	B	6.13%
deny	300	50	350	4811	0.103%	3.02%	B	24.58%
enlarge	300	50	350	529	0.011%	0.33%	C	4.69%
enlist	300	47	347	347	0.007%	0.22%	C	3.08%
forge	350	50	400	511	0.011%	0.32%	C	4.53%
furnish	300	50	350	396	0.008%	0.25%	C	3.51%
hail	300	50	350	775	0.017%	0.49%	C	6.88%
halt	250	50	300	856	0.018%	0.54%	C	7.59%
part	300	50	350	380	0.008%	0.24%	C	3.37%
plough	250	50	300	357	0.008%	0.22%	C	3.17%
plug	300	46	346	346	0.007%	0.22%	C	3.07%
pour	300	50	350	914	0.020%	0.57%	C	8.11%
say	500	50	550	94608	2.025%	59.31%	A	73.52%
smash	300	50	350	530	0.011%	0.33%	C	4.70%
smell	300	50	350	416	0.009%	0.26%	C	3.69%
steer	300	50	350	432	0.009%	0.27%	C	3.83%
submit	250	50	300	2258	0.048%	1.42%	B	11.54%
swell	300	50	350	398	0.009%	0.25%	C	3.53%
tell	500	50	550	21550	0.461%	13.51%	A	16.75%
throw	1000	50	1050	3710	0.079%	2.33%	B	18.96%
trouble	300	50	350	375	0.008%	0.24%	C	3.33%
wake	300	50	350	909	0.019%	0.57%	C	8.06%
yield	300	50	350	1482	0.032%	0.93%	B	7.57%
Celkom	10150	1493	11643	159518	3.414%	100.00%		300.00%

Tabuľka D.1: Základné charakteristiky 30 cieľových slovies.

D.2 Zmena charakteristík VPS po aplikovaní prahovej frekvencie

				Pred prah. frek.			Po prah. frek.		
Sloveso	**Skupina**	**Váha v skupine**	**Počet inštancií**	**Počet patternov**	**Priemerný počet inštancií na pattern**	**Perplexita**	**Počet patternov**	**Priemerný počet inštancií na pattern**	**Perplexita**
access	C	4.04%	300	10	30	3.670	4	75	3.143
ally	C	3.42%	250	8	31	4.390	5	50	3.921
arrive	B	31.22%	250	7	36	2.504	6	42	2.965
breathe	C	8.43%	350	18	19	7.748	7	50	5.084
claim	A	9.73%	500	11	45	2.986	6	83	2.958
cool	C	5.10%	300	16	19	8.806	7	43	5.511
crush	C	3.86%	350	14	25	8.860	9	39	6.926
cry	B	6.13%	250	15	17	4.995	5	50	3.532
deny	B	24.58%	300	12	25	6.716	7	43	5.173
enlarge	C	4.69%	300	6	50	1.450	5	60	2.290
enlist	C	3.08%	300	6	50	3.303	5	60	3.493
forge	C	4.53%	350	14	25	9.559	9	39	7.373
furnish	C	3.51%	300	9	33	5.427	5	60	4.295
hail	C	6.88%	300	10	30	3.317	5	60	2.892
halt	C	7.59%	250	4	63	0.405	4	63	1.806
part	C	3.37%	300	13	23	7.786	9	33	6.179
plough	C	3.17%	250	18	14	6.791	9	28	6.859
plug	C	3.07%	300	14	21	10.654	11	27	8.653
pour	C	8.11%	300	22	14	13.007	10	30	7.901
say	A	73.52%	500	16	31	1.201	6	83	1.906
smash	C	4.70%	300	12	25	5.314	6	50	3.967
smell	C	3.69%	300	11	27	7.062	8	38	5.844
steer	C	3.83%	300	24	13	17.271	14	21	11.105
submit	B	11.54%	250	6	42	1.988	5	50	2.627
swell	C	3.53%	300	25	12	15.909	11	27	9.041
tell	A	16.75%	500	18	28	4.159	9	56	3.767
throw	B	18.96%	1000	74	14	27.983	26	38	16.918
trouble	C	3.33%	300	14	21	0.117	10	30	6.150
wake	C	8.06%	300	11	27	5.671	7	43	4.765
yield	B	7.57%	300	12	25	8.926	10	30	7.371
A		80.66%	500	16	32	1.870	7	79	2.320
B		12.27%	408	22	28	8.949	10	42	6.482
C		7.07%	301	13	28	6.850	7	45	5.342
Priemer celkom			475	16	31	3.090	7	72	3.044

Tabuľka D.2: Zmena vybraných charakteristík slovies po aplikovaní prahovej frekvencie.

D.3 Zmena IAA po aplikovaní prahovej frekvencie

			Pred prah. frek.		Po prah. frek.	
Sloveso	**Skupina**	**Váha v skupine**	**Fleiss kappa**	**ARG**	**Fleiss kappa**	**ARG**
access	C	4.04%	0.683	0.762	0.729	0.943
ally	C	3.42%	0.733	1.229	0.73	1.217
arrive	B	31.22%	0.917	1.421	0.917	1.452
breathe	C	8.43%	0.835	1.309	0.839	1.378
claim	A	9.73%	0.840	1.303	0.845	1.303
cool	C	5.10%	0.875	1.846	0.875	1.846
crush	C	3.86%	0.634	0.748	0.654	0.942
cry	B	6.13%	0.826	1.221	0.843	1.354
deny	B	24.58%	0.691	1.145	0.689	1.138
enlarge	C	4.69%	0.623	0.646	0.623	0.646
enlist	C	3.08%	0.859	1.417	0.859	1.417
forge	C	4.53%	0.634	1.069	0.643	1.179
furnish	C	3.51%	0.798	1.251	0.797	1.204
hail	C	6.88%	0.835	0.943	0.834	0.937
halt	C	7.59%	0.697	0.521	0.697	0.509
part	C	3.37%	0.864	2.068	0.863	1.984
plough	C	3.17%	0.953	2.803	0.953	2.803
plug	C	3.07%	0.722	1.336	0.721	1.299
pour	C	8.11%	0.745	1.601	0.74	1.456
say	A	73.52%	0.876	0.801	0.903	0.875
smash	C	4.70%	0.787	1.267	0.812	1.331
smell	C	3.69%	0.882	1.716	0.882	1.716
steer	C	3.83%	0.722	1.195	0.732	1.509
submit	B	11.54%	0.879	0.879	0.879	0.879
swell	C	3.53%	0.804	1.518	0.816	1.789
tell	A	16.75%	0.915	1.381	0.932	1.583
throw	B	18.96%	0.613	0.721	0.645	1.026
trouble	C	3.33%	0.763	1.525	0.763	1.525
wake	C	8.06%	0.773	1.121	0.777	1.189
yield	B	7.57%	0.755	1.107	0.755	1.129
A		80.66%	0.879	0.947	0.902	1.035
B		12.27%	0.782	1.122	0.788	1.197
C		7.07%	0.782	1.122	0.788	1.197
Priemer celkom			0.860	0.981	0.880	1.067

Tabuľka D.3: Porovnanie dvoch mier medzianotátorskej zhody pred a po aplikovaní prahovej frekvencie.

D.4 Pokrytie subjektov vybranými segmentami LSP

Sloveso	Počet inštancií	Počet subjektov	Počet pasívnych	Počet os. zámen	Počet nom. subjektov	Počet nájdených subjektov	Pokrytie lexikónu	Inštancií so subjektom	Klasifikovaných inštancií
access	300	108	41	17	50	44	88.00%	36.00%	34.00%
ally	250	161	12	16	133	105	78.95%	64.40%	53.20%
arrive	250	184	3	48	133	117	87.97%	73.60%	67.20%
breathe	350	159	4	73	82	69	84.15%	45.43%	41.71%
claim	500	404	33	104	267	194	72.66%	80.80%	66.20%
cool	300	137	25	28	84	80	95.24%	45.67%	44.33%
crush	350	201	119	23	59	46	77.97%	57.43%	53.71%
cry	250	152	1	91	60	49	81.67%	60.80%	56.40%
deny	300	221	48	40	133	85	63.91%	73.67%	57.67%
enlarge	300	164	24	27	113	93	82.30%	54.67%	48.00%
enlist	300	190	16	50	124	96	77.42%	63.33%	54.00%
forge	350	187	78	31	78	66	84.62%	53.43%	50.00%
furnish	300	153	44	21	88	76	86.36%	51.00%	47.00%
hail	300	223	103	16	104	81	77.88%	74.33%	66.67%
halt	250	152	62	12	78	60	76.92%	60.80%	53.60%
part	300	152	26	54	72	64	88.89%	50.67%	48.00%
plough	250	121	38	26	57	50	87.72%	48.40%	45.60%
plug	300	167	18	56	93	67	72.04%	55.67%	47.00%
pour	300	210	34	37	139	111	79.86%	70.00%	60.67%
say	500	400	24	138	238	176	73.95%	80.00%	67.60%
smash	300	193	53	43	97	85	87.63%	64.33%	60.33%
smell	300	184	0	110	74	60	81.08%	61.33%	56.67%
steer	300	162	12	43	107	82	76.64%	54.00%	45.67%
submit	250	165	45	34	86	67	77.91%	66.00%	58.40%
swell	300	185	18	17	150	136	90.67%	61.67%	57.00%
tell	500	420	75	118	227	166	73.13%	84.00%	71.80%
throw	1000	649	168	155	326	258	79.14%	64.90%	58.10%
trouble	300	189	37	49	103	81	78.64%	63.00%	55.67%
wake	300	181	29	95	57	43	75.44%	60.33%	55.67%
yield	300	212	2	21	189	153	80.95%	70.67%	58.67%
Celkom	10150	6386	1192	1593	3601	2860	79.42%	62.92%	55.62%

Tabuľka D.4: Pokrytie subjektov kolekcie VPS lexikónom LSP s najvýhodnejšou konfiguráciou segmentov a použitých sémantických značiek.

D.5 Pokrytie objektov vybranými segmentami LSP

Sloveso	Počet inštancií	Počet objektov	Počet os. zámen	Počet nom. objektov	Počet nájdených objektov	Pokrytie lexikónu	Inštancií so subjektom	Klasifikovaných inštancií
access	300	155	0	155	124	80.00%	51.67%	41.33%
ally	250	83	6	77	29	37.66%	33.20%	14.00%
arrive	250	31	0	31	16	51.61%	12.40%	6.40%
breathe	350	124	0	124	98	79.03%	35.43%	28.00%
claim	500	402	0	402	221	54.98%	80.40%	44.20%
cool	300	122	0	122	82	67.21%	40.67%	27.33%
crush	350	260	10	250	192	76.80%	74.29%	57.71%
cry	250	51	2	49	30	61.22%	20.40%	12.80%
deny	300	261	0	261	167	63.98%	87.00%	55.67%
enlarge	300	158	0	158	123	77.85%	52.67%	41.00%
enlist	300	203	1	202	179	88.61%	67.67%	60.00%
forge	350	267	0	267	224	83.90%	76.29%	64.00%
furnish	300	211	9	202	145	71.78%	70.33%	51.33%
hail	300	204	7	197	134	68.02%	68.00%	47.00%
halt	250	204	1	203	163	80.30%	81.60%	65.60%
part	300	73	2	71	44	61.97%	24.33%	15.33%
plough	250	115	0	115	84	73.04%	46.00%	33.60%
plug	300	140	4	136	89	65.44%	46.67%	31.00%
pour	300	160	2	158	129	81.65%	53.33%	43.67%
say	500	392	0	392	185	47.19%	78.40%	37.00%
smash	300	215	2	213	172	80.75%	71.67%	58.00%
smell	300	145	1	144	88	61.11%	48.33%	29.67%
steer	300	175	7	168	127	75.60%	58.33%	44.67%
submit	250	170	2	168	130	77.38%	68.00%	52.80%
swell	300	103	0	103	81	78.64%	34.33%	27.00%
tell	500	428	141	287	190	66.20%	85.60%	66.20%
throw	1000	783	54	729	566	77.64%	78.30%	62.00%
trouble	300	127	26	101	59	58.42%	42.33%	28.33%
wake	300	108	19	89	49	55.06%	36.00%	22.67%
yield	300	222	1	221	159	71.95%	74.00%	53.33%
Celkom	10150	6092	297	5795	4079	70.39%	60.02%	43.11%

Tabuľka D.5: Pokrytie objektov kolekcie VPS lexikónom LSP s najvýhodnejšou konfiguráciou segmentov a použitých sémantických značiek.

D.6 Počet patternov v závislosti na prahovej frekvencii

Sloveso	**1**	**3**	**5**	**7**	**9**	**11**	**13**	**15**	**17**	**19**
access	10	9	8	4	4	4	4	4	4	4
ally	8	8	7	7	5	5	5	5	5	5
arrive	7	7	6	6	6	5	5	4	4	4
breathe	18	14	10	8	7	7	5	5	5	5
claim	11	10	9	7	6	5	4	4	4	4
cool	16	16	11	9	7	7	5	4	4	4
crush	14	12	11	10	9	7	7	7	7	6
cry	15	9	7	6	5	5	5	3	3	3
deny	12	10	10	8	7	7	6	6	6	6
enlarge	6	5	5	5	5	5	5	5	5	5
enlist	6	6	6	5	5	5	5	5	5	4
forge	14	14	12	10	9	9	9	9	9	8
furnish	9	9	7	6	5	5	5	5	5	5
hail	10	10	10	6	5	5	5	5	3	3
halt	5	5	4	4	4	4	4	4	3	3
part	13	12	12	10	9	8	8	8	8	5
plough	19	14	11	10	9	8	7	6	6	3
plug	14	13	13	12	11	10	9	9	8	8
pour	22	17	16	12	10	9	8	6	5	5
say	16	11	7	6	6	4	4	4	2	2
smash	12	11	10	7	6	5	5	5	4	4
smell	11	10	9	8	8	8	7	6	6	5
steer	24	24	20	18	14	10	9	9	5	4
submit	6	6	6	5	5	5	4	4	4	4
swell	25	22	19	13	11	10	9	8	8	6
tell	18	11	10	9	9	8	7	6	5	5
throw	74	60	49	37	26	22	20	20	17	15
trouble	15	13	11	10	10	7	6	6	5	4
wake	11	9	8	8	7	7	7	6	5	5
yield	12	12	11	11	10	7	7	6	6	6
Celkom	453	389	335	277	240	213	196	184	166	150
MAX	74	60	49	37	26	22	20	20	17	15
MIN	5	5	4	4	4	4	4	3	2	2

Tabuľka D.6: Počty patternov pre jednotlivé slovesá pre rôzne prahové frekvencie.

D.7 Zmena perplexity v závislosti na prahovej frekvencii

Verb	1	3	5	7	9	11	13	15	17	19
access	3.77	3.73	3.67	3.14	3.14	3.14	3.14	3.14	3.14	3.14
ally	4.27	4.27	4.23	4.23	3.92	3.92	3.92	3.92	3.92	3.92
arrive	2.99	2.99	2.97	2.97	2.97	2.86	2.86	2.65	2.65	2.65
breathe	6.89	6.57	5.88	5.41	5.08	5.08	4.26	4.26	4.26	4.26
claim	3.31	3.29	3.23	3.06	2.96	2.82	2.67	2.67	2.67	2.67
cool	7.82	7.82	6.91	6.32	5.51	5.51	4.51	3.97	3.97	3.97
crush	7.87	7.66	7.51	7.26	6.93	6.10	6.10	6.10	6.10	5.47
cry	4.71	4.30	3.98	3.78	3.53	3.53	3.53	2.78	2.78	2.78
deny	6.03	5.87	5.87	5.45	5.17	5.17	4.77	4.77	4.77	4.77
enlarge	2.30	2.29	2.29	2.29	2.29	2.29	2.29	2.29	2.29	2.29
enlist	3.52	3.52	3.52	3.49	3.49	3.49	3.49	3.49	3.49	3.34
forge	8.53	8.53	8.24	7.72	7.37	7.37	7.37	7.37	7.37	6.76
furnish	5.03	5.03	4.76	4.56	4.30	4.30	4.30	4.30	4.30	4.30
hail	3.53	3.53	3.53	3.08	2.89	2.89	2.89	2.89	2.32	2.32
halt	1.81	1.81	1.81	1.81	1.81	1.81	1.81	1.81	1.75	1.75
part	6.92	6.86	6.86	6.49	6.18	5.80	5.80	5.80	5.80	3.98
plough	8.97	8.48	7.70	7.33	6.86	6.26	5.63	4.94	4.94	2.85
plug	9.61	9.44	9.44	9.12	8.65	8.07	7.42	7.42	6.62	6.62
pour	12.18	11.35	11.05	9.11	7.90	7.20	6.48	5.00	4.28	4.28
say	2.14	2.09	1.96	1.91	1.91	1.75	1.75	1.75	1.52	1.52
smash	4.94	4.87	4.76	4.22	3.97	3.66	3.66	3.66	3.27	3.27
smell	6.31	6.21	6.07	5.84	5.84	5.84	5.48	5.03	5.03	4.47
steer	17.82	17.82	16.18	14.68	11.11	7.42	6.59	6.59	3.40	2.78
submit	2.66	2.66	2.66	2.63	2.63	2.63	2.50	2.50	2.50	2.50
swell	15.87	15.32	14.18	10.59	9.04	8.20	7.30	6.41	6.41	4.60
tell	4.11	3.93	3.86	3.77	3.77	3.63	3.46	3.26	3.02	3.02
throw	39.12	35.75	30.85	23.82	16.92	14.39	13.12	13.12	10.91	9.49
trouble	6.55	6.52	6.33	6.15	6.15	5.05	4.61	4.61	4.07	3.50
wake	5.21	5.10	4.97	4.97	4.77	4.77	4.77	4.42	3.99	3.99
yield	7.93	7.93	7.72	7.72	7.37	5.92	5.92	5.32	5.32	5.32

Tabuľka D.7: Zmena perplexity pre jednotlivé slovesá pre rôzne prahové frekvencie.

D.8 Počet inštancií so zmeneným patternom

Verb	**1**	**3**	**5**	**7**	**9**	**11**	**13**	**15**	**17**	**19**
access	0	2	5	26	26	26	26	26	26	26
ally	0	0	3	3	17	17	17	17	17	17
arrive	0	0	4	4	4	13	13	27	27	27
breathe	0	5	19	29	37	37	61	61	61	61
claim	0	1	5	16	23	33	44	44	44	44
cool	0	0	18	28	44	44	67	81	81	81
crush	0	4	7	13	21	40	40	40	40	58
cry	0	8	16	21	28	28	28	55	55	55
deny	0	3	3	14	21	21	33	33	33	33
enlarge	0	2	2	2	2	2	2	2	2	2
enlist	0	0	0	5	5	5	5	5	5	22
forge	0	0	7	18	25	25	25	25	25	42
furnish	0	0	7	13	21	21	21	21	21	21
hail	0	0	0	22	30	30	30	30	61	61
halt	0	0	3	3	3	3	3	3	18	18
part	0	2	2	13	21	30	30	30	30	82
plough	0	9	19	24	31	41	52	65	65	117
plug	0	2	2	7	15	25	36	36	51	51
pour	0	7	10	32	47	57	68	95	111	111
say	0	5	18	24	24	42	42	42	72	72
smash	0	2	6	22	30	40	40	40	55	55
smell	0	2	5	11	11	11	22	35	35	52
steer	0	0	12	24	54	93	104	104	166	184
submit	0	0	0	6	6	6	18	18	18	18
swell	0	5	15	46	62	72	84	97	97	131
tell	0	7	11	16	16	25	36	49	65	65
throw	0	23	61	125	206	244	266	266	312	347
trouble	0	2	8	13	13	43	54	54	70	88
wake	0	3	7	7	15	15	15	29	45	45
yield	0	0	4	4	12	41	41	54	54	54
Celkom	0	94	279	591	870	1130	1323	1484	1762	2040
MAX	0	23	61	125	206	244	266	266	312	347
MIN	0	0	0	2	2	2	2	2	2	2

Tabuľka D.8: Počet inštancií, ktorým sa zmení pattern po zavedení prahovej frekvencie.

D.9 Podiel patternu u v závislosti na prahovej frekvenci

Verb	1	3	5	7	9	11	13	15	17	19
access	1.0	1.7	2.7	9.7	9.7	9.7	9.7	9.7	9.7	9.7
ally	0.4	0.4	1.6	1.6	7.2	7.2	7.2	7.2	7.2	7.2
arrive	0.4	0.4	2.0	2.0	2.0	5.6	5.6	11.2	11.2	11.2
breathe	2.9	4.3	8.3	11.1	13.4	13.4	20.3	20.3	20.3	20.3
claim	2.4	2.6	3.4	5.6	7.0	9.0	11.2	11.2	11.2	11.2
cool	1.0	1.0	7.0	10.3	15.7	15.7	23.3	28.0	28.0	28.0
crush	1.7	2.9	3.7	5.4	7.7	13.1	13.1	13.1	13.1	18.3
cry	2.0	5.2	8.4	10.4	13.2	13.2	13.2	24.0	24.0	24.0
deny	2.3	3.3	3.3	7.0	9.3	9.3	13.3	13.3	13.3	13.3
enlarge	0.3	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
enlist	0.3	0.3	0.3	2.0	2.0	2.0	2.0	2.0	2.0	7.7
forge	1.1	1.1	3.1	6.3	8.3	8.3	8.3	8.3	8.3	13.1
furnish	3.3	3.3	5.7	7.7	10.3	10.3	10.3	10.3	10.3	10.3
hail	1.3	1.3	1.3	8.7	11.3	11.3	11.3	11.3	21.7	21.7
halt	0.0	0.0	1.2	1.2	1.2	1.2	1.2	1.2	7.2	7.2
part	0.7	1.3	1.3	5.0	7.7	10.7	10.7	10.7	10.7	28.0
plough	0.0	3.6	7.6	9.6	12.4	16.4	20.8	26.0	26.0	46.8
plug	3.3	4.0	4.0	5.7	8.3	11.7	15.3	15.3	20.3	20.3
pour	2.7	5.0	6.0	13.3	18.3	21.7	25.3	34.3	39.7	39.7
say	0.4	1.4	4.0	5.2	5.2	8.8	8.8	8.8	14.8	14.8
smash	1.7	2.3	3.7	9.0	11.7	15.0	15.0	15.0	20.0	20.0
smell	2.3	3.0	4.0	6.0	6.0	6.0	9.7	14.0	14.0	19.7
steer	2.3	2.3	6.3	10.3	20.3	33.3	37.0	37.0	57.7	63.7
submit	0.4	0.4	0.4	2.8	2.8	2.8	7.6	7.6	7.6	7.6
swell	1.0	2.7	6.0	16.3	21.7	25.0	29.0	33.3	33.3	44.7
tell	1.2	2.6	3.4	4.4	4.4	6.2	8.4	11.0	14.2	14.2
throw	2.1	4.4	8.2	14.6	22.7	26.5	28.7	28.7	33.3	36.8
trouble	0.0	0.7	2.7	4.3	4.3	14.3	18.0	18.0	23.3	29.3
wake	1.3	2.3	3.7	3.7	6.3	6.3	6.3	11.0	16.3	16.3
yield	3.0	3.0	4.3	4.3	7.0	16.7	16.7	21.0	21.0	21.0

Tabuľka D.9: Podiel patternu u (v percentách) vzhľadom ku všetkým inštanciám pre jednotlivé slovesá pre rôzne prahové frekvencie.

D.10 Experiment Default-FS 149

						Zlepšenie		
Sloveso	**S.**	**DT**	**kNN**	**SVM**	**ADA**	**r**	**%**	**ERD**
access	C	75.7 ±4.2	74.0 ±4.4	77.0 ±2.2	75.0 ±4.4	30.0	63.9	56.6
ally	C	61.6 ±4.7	62.8 ±4.8	65.6 ±3.6	64.8 ±3.0	17.9	37.6	34.3
arrive	B	70.4 ±5.8	72.0 ±3.8	70.0 ±4.1	72.8 ±3.1	2.1	3.0	6.4
breathe	C	63.4 ±5.3	59.4 ±3.6	66.0 ±7.2	67.7 ±4.8	28.2	74.8	45.4
claim	A	80.8 ±2.7	79.4 ±2.3	82.6 ±3.7	81.4 ±2.7	14.8	21.8	45.9
cool	C	59.0 ±6.8	59.3 ±6.5	62.4 ±4.6	63.3 ±8.1	35.1	128.8	48.3
crush	C	39.7 ±3.1	38.6 ±3.5	40.6 ±3.3	39.7 ±2.5	11.1	37.9	15.8
cry	B	69.6 ±8.9	68.7 ±6.4	74.8 ±3.2	75.2 ±3.5	22.5	42.9	47.1
deny	B	54.0 ±6.0	55.7 ±4.1	55.3 ±4.1	53.7 ±3.9	11.0	24.8	19.8
enlarge	C	81.7 ±1.7	79.4 ±3.6	82.7 ±3.1	82.4 ±3.6	6.1	7.9	25.9
enlist	C	72.7 ±5.3	74.3 ±4.0	74.0 ±3.8	76.7 ±5.2	25.0	51.1	49.1
forge	C	48.0 ±4.0	41.7 ±5.8	49.5 ±4.2	47.7 ±3.0	23.2	88.1	31.4
furnish	C	66.0 ±6.9	61.3 ±4.0	67.7 ±5.8	66.0 ±5.8	24.0	55.1	42.7
hail	C	80.7 ±2.9	82.7 ±2.1	82.7 ±2.7	82.0 ±2.7	15.3	22.7	46.9
halt	C	85.6 ±3.0	84.4 ±1.9	86.0 ±2.9	84.8 ±5.3	2.4	2.9	14.7
part	C	71.7 ±4.3	70.0 ±5.0	74.7 ±3.0	74.3 ±4.4	31.6	73.3	55.5
plough	C	64.8 ±6.0	56.5 ±4.7	68.5 ±3.6	69.6 ±4.2	35.9	110.6	53.3
plug	C	53.0 ±6.0	49.4 ±6.3	51.7 ±5.7	53.7 ±4.4	20.4	65.1	29.7
pour	C	48.6 ±5.5	49.3 ±3.7	53.4 ±8.0	47.3 ±8.1	28.9	118.5	38.3
say	A	88.0 ±0.7	89.2 ±1.8	89.8 ±2.2	89.4 ±2.4	4.6	5.4	31.2
smash	C	61.7 ±2.2	62.4 ±4.0	64.7 ±2.5	64.4 ±5.0	11.3	21.2	24.2
smell	C	54.7 ±3.2	54.4 ±4.1	58.3 ±4.8	55.7 ±3.5	22.0	60.6	34.6
steer	C	42.4 ±5.3	38.4 ±5.6	43.7 ±5.9	42.4 ±4.9	26.0	147.5	31.6
submit	B	87.6 ±4.0	82.0 ±5.8	86.0 ±2.9	86.0 ±4.3	15.3	21.6	52.2
swell	C	38.7 ±5.1	40.3 ±7.8	45.0 ±3.4	42.0 ±6.1	28.0	165.2	33.8
tell	A	72.8 ±2.7	71.6 ±2.9	76.4 ±1.4	75.4 ±2.3	11.2	17.2	32.2
throw	B	42.6 ±3.5	35.1 ±4.2	42.3 ±3.2	45.0 ±4.2	19.6	86.3	25.4
trouble	C	66.7 ±5.0	63.3 ±4.8	68.4 ±4.3	68.3 ±4.3	24.0	54.0	43.1
wake	C	75.4 ±4.6	68.7 ±3.6	76.0 ±2.4	76.7 ±3.9	31.0	68.9	56.4
yield	B	42.0 ±6.7	39.4 ±5.7	43.6 ±3.8	42.3 ±7.8	15.3	53.9	21.3
A		84.8 ±1.2	85.3 ±2.1	86.9 ±2.2	86.3 ±2.5	6.7	9.0	32.8
B		60.9 ±5.5	59.5 ±4.5	61.3 ±3.7	62.2 ±4.0	11.4	32.6	22.2
C		63.9 ±4.4	62.0 ±4.2	66.1 ±4.3	65.3 ±4.7	22.4	67.5	38.7
Celkom		80.4 ±2.0	80.5 ±2.5	82.3 ±2.5	81.8 ±2.8	8.4	16.0	31.9

Tabuľka D.10: Výsledky experimentu **Default-FS 149**. Modely klasifikátorov boli natrénované pomocou všetkých dostupných morfologicko-syntaktických rysov. Najlepším algoritmom je SVM, hoci medzi výsledkami algoritmov neexistuje štatisticky signifikantný rozdiel.

D.11 Rebríček A

P.	Rys	P.	Rys	P.	Rys	P.	Rys
1	negation	39	1n_nominal	77	prep_upon	115	prt_in
2	3p_be	40	tense_vbn	78	tmod	116	prepc_without
3	advmod	41	tense_vbp	79	prepc_other	117	prep_as
4	3p_modal	42	3n_nominal	80	2n_modal	118	infinitive
5	prt_off	43	2n_nominal	81	prep_without	119	prep_other
6	prt_down	44	any_obj	82	nsubjpass	120	2p_modal
7	1p_adverbial	45	3n_verbs	83	prepc_none	121	2p_nominal
8	1p_wh-pronoun	46	3n_adverbial	84	prep_beyond	122	prt_out
9	2n_verbs	47	mark_since	85	1p_nominal	123	prep_on
10	2n_be	48	prt_up	86	prep_against	124	3p_wh-adverb
11	prep_by	49	3p_adjective	87	tense_vbd	125	mark_while
12	tense_vbg	50	2n_adjective	88	prep_around	126	subj_pl
13	1p_adjective	51	prep_at	89	3n_be	127	1p_wh-adverb
14	prep_behind	52	complm	90	prt_back	128	3p_nominal
15	mark_none	53	1p_modal	91	prep_under	129	prt_other
16	n_subj	54	2p_wh-adverb	92	prep_out	130	prep_out_of
17	prepc_after	55	passive_voice	93	modality_2	131	prepc_in
18	3n_to	56	1n_be	94	3n_wh-pronoun	132	2p_adjective
19	2n_to	57	obj_pl	95	1n_wh-adverb	133	2p_verbs
20	2p_wh-pronoun	58	1n_verbs	96	prep_in	134	prep_before
21	prep_during	59	1p_verbs	97	1n_adverbial	135	advcl
22	c_subj	60	prep_towards	98	prt_on	136	mark_in
23	prep_for	61	modality_1	99	prep_within	137	prep_between
24	prep_into	62	prepc_by	100	1p_to	138	3n_modal
25	mark_other	63	1n_modal	101	2n_wh-adverb	139	prt_over
26	purpcl	64	3p_to	102	mark_because	140	prepc_with
27	prepc_according_to	65	1n_wh-pronoun	103	prep_to	141	tense_vb
28	prep_none	66	mark_after	104	2p_be	142	prep_through
29	1n_adjective	67	2n_adverbial	105	mark_if	143	iobj
30	prepc_as	68	3n_adjective	106	prep_like	144	prep_over
31	3p_adverbial	69	prep_away_from	107	prep_than	145	prep_across
32	mark_until	70	1p_be	108	2n_wh-pronoun	146	prepc_into
33	prep_after	71	prepc_for	109	prep_despite	147	2p_to
34	dobj	72	mark_before	110	mark_as	148	prep_about
35	prep_of	73	3p_wh-pronoun	111	3p_verbs	149	csubjpass
36	1n_to	74	prt_none	112	ccomp		
37	2p_adverbial	75	prt_away	113	3n_wh-adverb		
38	prep_with	76	prep_from	114	mark_although		

Tabuľka D.11: **Rebríček A**. Morfologicko-syntaktické rysy zoradené podľa úspešnosti binárnych klasifikátorov s použitím jedného rysu.

D.12 Experiment A20

						Zlepšenie		
Sloveso	**S.**	**DT**	**kNN**	**SVM**	**ADA**	**r**	**%**	**ERD**
access	C	61.0 ±5.8	60.1 ±7.7	60.7 ±7.4	61.0 ±8.5	13.4	28.4	25.2
ally	C	49.2 ±3.6	47.2 ±4.0	51.2 ±4.4	46.3 ±3.6	3.5	7.3	6.7
arrive	B	68.0 ±1.4	70.4 ±3.2	68.8 ±2.1	69.6 ±3.2	0.9	1.3	2.7
breathe	C	47.7 ±3.4	46.9 ±1.8	48.3 ±3.6	48.0 ±3.3	10.5	27.9	16.9
claim	A	72.4 ±2.1	72.2 ±2.1	72.0 ±2.6	73.2 ±1.9	4.2	6.2	13.0
cool	C	46.0 ±5.9	45.7 ±5.4	47.0 ±6.0	45.4 ±5.7	19.8	72.5	27.2
crush	C	31.7 ±2.3	31.1 ±3.1	31.4 ±2.6	31.7 ±2.1	2.0	6.8	2.8
cry	B	53.2 ±2.5	52.4 ±7.9	54.8 ±3.7	53.6 ±2.5	2.5	4.7	5.2
deny	B	44.7 ±1.6	44.7 ±3.1	45.1 ±4.5	44.7 ±2.6	0.7	1.6	1.3
enlarge	C	76.7 ±1.7	76.7 ±1.7	77.4 ±2.5	82.3 ±2.1	0.7	0.9	3.0
enlist	C	51.3 ±1.1	51.7 ±2.8	50.3 ±1.3	50.7 ±1.4	1.3	2.7	2.6
forge	C	28.3 ±5.8	26.9 ±5.5	28.9 ±4.5	27.8 ±6.2	2.6	9.8	3.5
furnish	C	48.0 ±4.4	48.7 ±3.9	49.3 ±4.0	47.7 ±4.6	5.7	13.0	10.1
hail	C	67.4 ±1.8	68.3 ±1.5	67.7 ±1.6	67.7 ±2.3	0.3	0.4	0.9
halt	C	83.6 ±1.3	84.0 ±2.8	83.6 ±1.3	83.6 ±1.3	0.0	0.0	0.1
part	C	44.0 ±2.2	43.4 ±2.2	44.7 ±2.5	43.4 ±2.0	1.6	3.7	2.8
plough	C	40.8 ±4.0	42.0 ±3.9	41.7 ±5.7	42.1 ±5.2	9.2	28.2	13.6
plug	C	34.0 ±2.5	36.7 ±6.2	37.7 ±4.4	36.4 ±5.0	6.4	20.3	9.3
pour	C	26.3 ±4.1	27.0 ±5.7	27.0 ±3.0	26.3 ±4.1	2.6	10.6	3.4
say	A	86.2 ±1.4	85.8 ±1.1	85.2 ±0.6	86.2 ±1.4	0.0	0.0	0.1
smash	C	53.4 ±1.3	53.0 ±1.9	54.7 ±2.9	53.0 ±1.5	1.3	2.4	2.7
smell	C	39.3 ±3.5	38.7 ±5.6	38.0 ±5.7	38.0 ±3.9	0.4	1.0	0.6
steer	C	20.3 ±1.0	20.7 ±3.7	21.3 ±3.0	19.7 ±3.2	3.7	20.7	4.4
submit	B	70.8 ±0.9	70.8 ±0.9	70.8 ±2.7	70.0 ±1.9	0.1	0.1	0.3
swell	C	31.7 ±6.5	30.1 ±6.7	29.0 ±4.0	27.7 ±3.3	12.1	71.1	14.5
tell	A	70.8 ±1.5	69.6 ±1.4	70.8 ±1.5	70.4 ±1.7	5.6	8.6	16.1
throw	B	26.6 ±1.2	25.7 ±2.3	27.2 ±2.4	26.7 ±0.9	4.5	19.8	5.8
trouble	C	49.3 ±1.7	51.0 ±3.7	52.0 ±5.0	51.3 ±4.2	7.6	17.2	13.7
wake	C	59.1 ±4.7	54.0 ±2.9	56.7 ±5.1	58.7 ±5.9	11.7	26.1	21.3
yield	B	40.3 ±5.2	40.0 ±4.9	38.3 ±5.1	37.3 ±5.2	10.0	35.2	13.9
A		82.3 ±1.5	81.8 ±1.3	81.5 ±0.9	82.3 ±1.5	1.4	2.1	4.1
B		51.7 ±1.7	52.2 ±3.1	52.2 ±3.2	52.0 ±2.6	2.2	7.5	3.7
C		49.1 ±3.3	48.7 ±3.7	49.4 ±3.7	49.1 ±3.8	5.6	17.4	9.1
Celkom		76.2 ±1.7	75.8 ±1.7	75.6 ±1.4	76.2 ±1.8	1.8	3.8	4.4

Tabuľka D.12: Výsledky experimentu **A20**. Modely klasifikátorov boli natrénované pomocou 20 najlepších rysov podľa rebríčka A. Najlepším algoritmom je SVM, hoci medzi výsledkami algoritmov neexistuje štatisticky signifikantný rozdiel.

D.13 Rebríček W

P.	Rys	P.	Rys	P.	Rys	P.	Rys
1	any_obj	25	2n_adjective	49	prt_up	73	2p_modal
2	obj_pl	26	subj_pl	50	prep_out_of	74	prep_against
3	1n_to	27	1p_be	51	passive_voice	75	prep_like
4	dobj	28	advmod	52	prt_away	76	3p_adjective
5	tense_vbg	29	complm	53	prep_of	77	prep_from
6	n_subj	30	mark_none	54	2n_verbs	78	prt_in
7	1n_adjective	31	prep_none	55	prep_with	79	prepc_into
8	3p_be	32	prep_on	56	prt_back	80	3p_wh.adverb
9	1n_adverbial	33	advcl	57	prep_about	81	prep_as
10	prep_to	34	2n_adverbial	58	1p_to	82	prepc_as
11	2n_to	35	prt_down	59	2p_verbs	83	prepc_none
12	tense_vbn	36	2p_adverbial	60	prt_other	84	2p_to
13	1p_modal	37	3n_verbs	61	nsubjpass	85	prep_between
14	3n_nominal	38	prep_by	62	3n_to	86	prep_other
15	tense_vb	39	prep_into	63	ccomp	87	prep_upon
16	1n_nominal	40	1p_verbs	64	infinitive	88	1n_verbs
17	tense_vbd	41	3n_adjective	65	3p_adverbial	89	negation
18	2n_nominal	42	prep_at	66	modality_2	90	3n_be
19	1p_wh.pronoun	43	3p_nominal	67	2p_be	91	iobj
20	2p_nominal	44	3p_verbs	68	2p_adjective	92	2n_be
21	3n_adverbial	45	prt_out	69	prep_away_from	93	1p_adjective
22	1p_nominal	46	prep_in	70	tense_vbp	94	c_subj
23	1p_adverbial	47	prt_off	71	prep_through	95	2p_wh.adverb
24	prep_for	48	prep_behind	72	prt_none		

Tabuľka D.13: **Rebríček W**. Morfologicko-syntaktické rysy zoradené podľa úspešnosti binárnych klasifikátorov s použitím všetkých rysov.

D.14 Experiment W20

						Zlepšenie		
Sloveso	**S.**	**DT**	**kNN**	**SVM**	**ADA**	**r**	**%**	**ERD**
access	C	76.7 ±3.2	76.3 ±4.6	77.3 ±3.3	76.7 ±3.2	30.3	64.6	57.2
ally	C	62.8 ±3.4	62.8 ±6.7	64.4 ±4.6	63.6 ±7.0	16.7	35.0	31.9
arrive	B	68.4 ±1.6	69.6 ±3.9	70.8 ±3.9	68.0 ±4.5	0.8	1.2	2.6
breathe	C	59.4 ±3.9	56.3 ±5.1	60.3 ±5.7	59.7 ±5.0	22.5	59.7	36.2
claim	A	81.4 ±3.4	79.2 ±4.7	81.6 ±3.4	80.8 ±3.0	13.8	20.4	42.9
cool	C	63.3 ±6.5	62.0 ±6.0	64.0 ±7.4	61.3 ±7.0	36.7	134.8	50.5
crush	C	38.0 ±4.6	41.4 ±4.4	41.7 ±3.9	40.3 ±3.7	12.3	41.8	17.4
cry	B	62.1 ±6.5	62.0 ±6.1	64.8 ±4.8	60.9 ±6.6	10.9	20.8	22.8
deny	B	51.4 ±4.4	51.7 ±4.1	55.0 ±2.1	52.0 ±3.8	10.7	24.1	19.2
enlarge	C	77.7 ±2.7	80.4 ±1.9	80.7 ±2.5	82.3 ±2.1	4.0	5.3	17.3
enlist	C	71.4 ±4.7	72.4 ±6.1	72.7 ±3.6	72.0 ±5.2	23.7	48.4	46.5
forge	C	38.6 ±5.2	33.5 ±4.6	38.0 ±5.0	38.3 ±5.4	11.7	44.7	15.9
furnish	C	58.4 ±5.3	56.1 ±6.9	60.0 ±4.6	58.8 ±7.0	16.4	37.5	29.0
hail	C	77.0 ±3.7	78.0 ±3.7	77.3 ±4.2	79.7 ±3.3	10.0	14.8	30.5
halt	C	83.6 ±1.3	84.0 ±3.2	84.8 ±3.0	84.0 ±3.1	1.2	1.4	7.3
part	C	72.3 ±3.9	71.0 ±2.8	73.0 ±3.6	70.4 ±4.2	29.9	69.5	52.6
plough	C	53.2 ±3.6	50.8 ±4.4	68.5 ±3.6	69.6 ±4.2	35.9	110.6	53.3
plug	C	42.7 ±4.3	42.1 ±5.3	47.7 ±5.2	42.4 ±4.0	16.4	52.3	23.8
pour	C	44.0 ±5.1	41.7 ±8.4	45.1 ±5.1	44.0 ±3.3	20.6	84.5	27.3
say	A	88.2 ±1.7	88.2 ±2.1	89.2 ±2.2	88.2 ±1.4	4.0	4.7	27.1
smash	C	60.4 ±3.6	60.4 ±2.5	64.4 ±2.9	60.7 ±2.7	11.0	20.5	23.5
smell	C	56.7 ±3.3	53.7 ±4.4	55.0 ±4.2	56.4 ±4.1	18.7	51.5	29.4
steer	C	36.7 ±4.8	35.4 ±7.2	39.7 ±6.1	37.7 ±5.8	22.1	124.9	26.8
submit	B	86.0 ±3.0	84.4 ±4.1	86.0 ±4.5	85.6 ±4.4	15.3	21.6	52.2
swell	C	38.7 ±5.1	38.7 ±3.4	42.6 ±5.7	39.4 ±5.1	25.7	151.2	30.9
tell	A	70.4 ±3.3	71.2 ±1.8	72.6 ±3.8	71.0 ±2.6	7.4	11.4	21.3
throw	B	27.7 ±4.3	26.6 ±2.6	27.6 ±4.2	29.5 ±4.3	4.9	21.6	6.3
trouble	C	65.0 ±4.0	60.7 ±4.7	67.4 ±3.9	64.4 ±4.5	22.6	51.0	40.7
wake	C	75.4 ±4.6	77.7 ±3.5	79.7 ±4.4	76.7 ±4.1	34.7	77.1	63.0
yield	B	43.7 ±6.2	40.3 ±5.8	45.0 ±6.6	42.3 ±4.2	16.7	58.8	23.3
A		84.6 ±2.1	84.5 ±2.3	85.7 ±2.6	84.6 ±1.8	5.5	7.4	27.7
B		56.3 ±3.6	56.1 ±4.0	58.2 ±3.8	56.4 ±4.4	7.5	18.6	15.9
C		61.2 ±4.1	60.5 ±4.7	63.4 ±4.4	62.3 ±4.3	19.7	58.7	33.5
Celkom		79.4 ±2.4	79.3 ±2.7	80.7 ±2.9	79.6 ±2.3	6.8	12.4	26.6

Tabuľka D.14: Výsledky experimentu **W20**. Modely klasifikátorov boli natrénované pomocou 20 najlepších rysov podľa rebríčka W. Najlepším algoritmom je SVM, hoci medzi výsledkami algoritmov neexistuje štatisticky signifikantný rozdiel.

D.15 Rebríček G

P.	Rys	P.	Rys	P.	Rys	P.	Rys
1	any_obj	39	X1n_adverbial	77	X2n_wh.pronoun	115	prep_under
2	passive_voice	40	X2p_be	78	prt_away	116	prep_than
3	X1n_nominal	41	X1n_adjective	79	X2n_wh.adverb	117	prep_within
4	modality_1	42	X2n_adjective	80	X2n_be	118	prep_without
5	X1n_to	43	X1p_wh.pronoun	81	nsubjpass	119	prep_around
6	tense_vbn	44	X1p_be	82	X3n_verbs	120	prep_before
7	modality_2	45	prep_none	83	prt_up	121	prep_towards
8	tense_vbg	46	prep_to	84	X3n_modal	122	prep_upon
9	negation	47	X2p_modal	85	prep_against	123	prep_despite
10	X1p_to	48	prt_out	86	prep_away_from	124	prep_across
11	tense_vb	49	X2p_to	87	X3n_wh.pronoun	125	prep_beyond
12	tense_vbd	50	X2p_adverbial	88	prep_through	126	prep_out
13	prep_with	51	X2p_wh.pronoun	89	X3n_wh.adverb	127	prepc_other
14	prep_into	52	X2p_verbs	90	X3n_adverbial	128	prepc_none
15	obj_pl	53	X2p_wh.adverb	91	X3n_be	129	prepc_by
16	X1p_adverbial	54	X2p_nominal	92	prep_by	130	prepc_in
17	tense_vbp	55	dobj	93	prep_behind	131	prepc_with
18	prep_from	56	X1p_adjective	94	c_subj	132	prepc_as
19	n_subj	57	complm	95	prep_out_of	133	prepc_for
20	prep_on	58	X1p_modal	96	csubjpass	134	prepc_after
21	X3p_adjective	59	advmod	97	iobj	135	prepc_into
22	infinitive	60	X2n_nominal	98	subj_pl	136	prepc_without
23	X3p_modal	61	X1p_wh.adverb	99	prt_other	137	prepc_according_to
24	X2n_to	62	prep_for	100	prep_as	138	mark_none
25	prep_at	63	prep_about	101	prt_off	139	mark_other
26	ccomp	64	X2n_adverbial	102	prt_back	140	mark_as
27	X3p_verbs	65	X3n_to	103	prt_on	141	mark_if
28	X1n_verbs	66	X1n_modal	104	prt_in	142	mark_although
29	X3p_adverbial	67	X3n_nominal	105	prt_over	143	mark_because
30	X3p_be	68	X1n_wh.pronoun	106	purpcl	144	mark_while
31	prt_none	69	X1n_wh.adverb	107	tmod	145	mark_after
32	X3p_nominal	70	X1n_be	108	advcl	146	mark_before
33	X3p_to	71	prt_down	109	prep_other	147	mark_until
34	X2p_adjective	72	prep_of	110	prep_in	148	mark_since
35	X3p_wh.pronoun	73	X2n_modal	111	prep_after	149	mark_in
36	X1p_nominal	74	X2n_verbs	112	prep_over		
37	X1p_verbs	75	prep_between	113	prep_during		
38	X3p_wh.adverb	76	X3n_adjective	114	prep_like		

Tabuľka D.15: **Rebríček G**. Morfologicko-syntaktické rysy zoradené podľa úspešnosti binárnych klasifikátorov s použitím hladového algortmu pri výbere rysov.

D.16 Najlepšie morfo-syntaktické rysy podľa hladového algoritmu

Sloveso	Najúspešnejšie rysy
access	X1n_to, X1p_nominal, any_obj,
ally	X1n_nominal, tense_vb, tense_vbg, passive_voice
arrive	X1p_to, prep_at, X1p_verbs, advmod, X1p_nominal,
breathe	X1p_adverbial, prep_into, n_subj, tense_vb, tense_vbg, prt_out, any_obj, X3p_nominal, X3p_verbs,
claim	X1n_to,
cool	any_obj, n_subj, X1p_to, dobj, tense_vbn, nsubjpass, X1p_verbs,
crush	obj_pl, tense_vbg, X1p_adverbial, X1n_nominal, tense_vbn,
cry	prt_none, prep_for, ccomp, X1n_nominal, X2n_nominal,
deny	X1n_verbs, prep_to, prep_on, X1n_adjective
enlarge	prep_on,
enlist	any_obj, X1p_be, obj_pl, dobj
forge	prep_with, complm, prep_between, prep_none, X2n_adjective, tense_vbg, any_obj, X2n_nominal, X2p_nominal, advmod, n_subj, X1p_adverbial
furnish	prep_with, X2p_be, tense_vbn, X1p_adverbial, X1p_verbs,
hail	prep_from,
halt	passive_voice,
part	X1n_nominal, prep_from, X1n_adverbial, tense_vbn, advmod, X2n_nominal, tense_vbd,
plough	any_obj, prep_into, prt_none, tense_vb, prep_through, X1n_nominal, X1p_be,
plug	X1p_to, X1p_wh.pronoun, X2n_adjective, prep_none, tense_vbg,
pour	any_obj, prt_out, prep_from, X1n_nominal, prt_away,
say	X1n_to, modality_2
smash	prep_into, X1n_nominal, X3p_be, prep_against
smell	X1n_nominal, prep_none, n_subj, X3n_nominal
steer	any_obj, X1n_adjective, X2n_adjective, prep_away_from, tense_vbg, X2n_adverbial
submit	ccomp, X1n_adverbial
swell	obj_pl, tense_vbn, tense_vbg, prep_to, X1p_verbs,
tell	X2n_to, prep_about, prep_of, X3n_to, X1n_to, X3p_be
throw	prep_at, prep_into, prep_on, prt_down, prep_behind, prep_out_of, prt_up, prt_out, X2n_adverbial, tense_vbn
trouble	X1n_to, prep_with
wake	tense_vbn, X1n_nominal, tense_vbg, X2n_to
yield	any_obj, tense_vb, X2p_adjective, tense_vbn, prep_to, n_subj

Tabuľka D.16: Najúspešnejšie rysy vybrané hladovým algoritmom individuálne pre každé sloveso. Hladový algoritmus skončil s pridávaním rysov po dosiahnutí prvého (lokálneho) maxima v Accuracy.

D.17 Rebríček Best58

P.	Rys	P.	Rys	P.	Rys	P.	Rys
1	any_obj	16	ccomp	31	X3p_be	46	X3n_to
2	X1n_nominal	17	prt_none	32	X2p_be	47	prep_away_from
3	X1n_to	18	passive_voice	33	complm	48	X3n_nominal
4	tense_vbn	19	X2n_to	34	modality_2	49	prep_behind
5	prep_into	20	prep_none	35	X1p_wh.pronoun	50	prep_through
6	prep_with	21	X1n_verbs	36	dobj	51	prt_away
7	obj_pl	22	prep_to	37	prep_for	52	nsubjpass
8	X1p_to	23	X1p_verbs	38	prep_about	53	prep_out_of
9	tense_vbg	24	X2n_adjective	39	X2n_nominal	54	prt_up
10	prep_from	25	X1n_adverbial	40	prep_between	55	tense_vbd
11	X1p_adverbial	26	prt_out	41	X2p_adjective	56	X3p_nominal
12	prep_on	27	X1n_adjective	42	prep_of	57	X3p_verbs
13	prep_at	28	X1p_nominal	43	X2n_adverbial	58	X2p_nominal
14	tense_vb	29	X1p_be	44	prt_down		
15	n_subj	30	advmod	45	prep_against		

Tabuľka D.17: **Rebríček Best58**, tvorený 58 najlepšími morfo-syntaktickými rysmi, získanými hladovým algoritmom, ktorý skončil vyberanie rysov po prvom poklese Accuracy.

D.18 Experiment Best58

						Zlepšenie		
Sloveso	**S.**	**DT**	**kNN**	**SVM**	**ADA**	**r**	**%**	**ERD**
access	C	75.7 ±4.2	74.7 ±4.7	78.0 ±3.0	75.3 ±3.8	31.0	66.0	58.5
ally	C	62.4 ±4.7	65.2 ±4.4	66.0 ±4.7	64.8 ±3.9	18.3	38.3	34.9
arrive	B	70.8 ±4.9	70.0 ±4.0	70.8 ±4.2	72.8 ±3.6	2.9	4.2	8.9
breathe	C	63.4 ±3.3	61.7 ±3.1	66.6 ±7.0	64.6 ±5.1	28.8	76.4	46.3
claim	A	80.6 ±2.7	79.4 ±1.4	81.6 ±2.4	80.4 ±2.9	13.8	20.3	42.8
cool	C	59.4 ±7.4	59.7 ±6.9	63.0 ±7.6	63.3 ±5.8	35.7	131.1	49.1
crush	C	42.5 ±4.5	38.3 ±4.4	42.3 ±3.7	40.6 ±4.4	12.9	43.7	18.2
cry	B	66.4 ±6.1	68.0 ±4.1	72.4 ±4.5	75.2 ±3.9	20.1	38.3	42.1
deny	B	56.4 ±5.1	54.7 ±4.5	56.7 ±2.2	55.0 ±4.1	12.3	27.8	22.2
enlarge	C	78.3 ±1.4	80.4 ±3.8	82.0 ±3.5	83.4 ±3.2	5.4	7.0	23.1
enlist	C	73.4 ±5.0	75.0 ±5.4	74.4 ±4.3	75.7 ±4.5	25.4	51.8	49.8
forge	C	47.4 ±3.7	43.5 ±5.2	48.6 ±3.8	47.7 ±4.4	22.3	84.8	30.2
furnish	C	67.0 ±5.6	63.0 ±4.5	66.0 ±5.3	67.0 ±4.7	22.4	51.2	39.7
hail	C	81.0 ±1.7	81.7 ±2.7	84.7 ±3.5	81.3 ±4.1	17.3	25.7	53.0
halt	C	84.1 ±4.1	85.2 ±1.6	85.2 ±2.3	84.8 ±2.6	1.2	1.4	7.3
part	C	71.0 ±4.0	69.6 ±5.2	74.0 ±3.5	76.0 ±5.1	30.9	71.8	54.3
plough	C	64.8 ±6.0	60.1 ±5.3	68.9 ±5.6	68.5 ±4.9	36.4	111.8	53.9
plug	C	50.0 ±5.4	48.4 ±6.0	50.3 ±3.1	49.7 ±4.5	19.0	60.8	27.7
pour	C	46.7 ±3.9	46.0 ±5.2	50.7 ±8.5	48.0 ±6.7	26.3	107.7	34.8
say	A	89.0 ±2.4	89.0 ±1.6	90.2 ±2.6	89.2 ±2.2	5.0	5.9	33.9
smash	C	61.7 ±2.2	61.7 ±3.9	64.4 ±7.7	63.7 ±4.1	11.0	20.7	23.7
smell	C	54.7 ±3.2	53.0 ±4.2	57.4 ±5.1	55.7 ±4.2	21.0	57.9	33.0
steer	C	42.1 ±5.2	37.7 ±3.9	40.0 ±3.2	40.7 ±5.1	22.4	126.6	27.2
submit	B	87.6 ±4.0	82.8 ±5.4	85.6 ±5.8	84.8 ±4.7	14.9	21.0	50.9
swell	C	38.7 ±5.1	41.6 ±4.9	47.0 ±7.8	40.4 ±8.6	30.1	177.1	36.2
tell	A	73.4 ±2.6	71.4 ±2.1	76.0 ±1.8	75.4 ±1.5	10.8	16.5	31.0
throw	B	40.8 ±2.6	39.4 ±4.1	43.2 ±3.2	43.9 ±3.8	20.4	89.9	26.4
trouble	C	66.7 ±3.4	64.0 ±5.0	69.7 ±3.2	66.7 ±4.9	25.3	57.1	45.5
wake	C	75.4 ±4.6	73.0 ±4.4	76.3 ±2.5	77.4 ±4.2	31.3	69.6	57.0
yield	B	44.0 ±7.3	40.0 ±3.3	46.7 ±6.1	41.3 ±6.6	18.3	64.6	25.5
A		85.6 ±2.4	85.1 ±1.7	87.0 ±2.4	86.0 ±2.2	6.8	9.1	34.3
B		61.2 ±4.7	59.5 ±4.3	62.1 ±3.8	62.1 ±4.1	12.1	34.9	23.6
C		63.6 ±4.1	62.6 ±4.3	66.0 ±4.8	64.9 ±4.7	22.2	66.5	38.2
Celkom		81.0 ±2.8	80.4 ±2.2	82.4 ±2.8	81.6 ±2.6	8.6	16.3	33.3

Tabuľka D.18: Výsledky experimentu **Best58**. Modely klasifikátorov boli natrénované pomocou 58 najlepších rysov podľa rebríčka Best58. Najlepším algoritmom je SVM, hoci medzi výsledkami algoritmov neexistuje štatisticky signifikantný rozdiel.

D.19 Experiment Default-FS+NER

						Zlepšenie		
Sloveso	**S.**	**DT**	**kNN**	**SVM**	**ADA**	**r**	**%**	**ERD**
access	C	75.7 ±4.2	74.3 ±4.5	76.7 ±2.8	74.7 ±4.2	29.7	63.2	56.0
ally	C	61.6 ±4.7	63.2 ±5.2	63.2 ±2.3	64.0 ±3.3	15.5	32.5	29.7
arrive	B	70.4 ±5.8	70.4 ±5.1	69.6 ±3.8	71.6 ±2.1	1.7	2.4	5.2
breathe	C	63.4 ±5.1	58.9 ±3.6	65.7 ±6.3	66.3 ±4.2	28.0	74.1	44.9
claim	A	79.6 ±2.4	78.8 ±3.1	82.8 ±3.7	81.8 ±2.8	15.0	22.1	46.5
cool	C	59.0 ±6.8	59.7 ±5.7	62.4 ±4.6	63.4 ±7.3	35.1	128.8	48.3
crush	C	39.7 ±3.9	37.4 ±4.0	42.3 ±4.0	40.0 ±3.6	12.9	43.7	18.2
cry	B	68.8 ±8.8	69.2 ±5.7	74.8 ±3.5	77.6 ±4.3	22.5	42.9	47.1
deny	B	53.0 ±6.5	57.1 ±5.2	55.0 ±4.4	51.4 ±4.2	10.7	24.0	19.1
enlarge	C	81.7 ±1.7	79.4 ±2.2	82.7 ±2.9	82.4 ±2.0	6.1	7.9	25.9
enlist	C	72.7 ±5.3	74.3 ±3.7	73.7 ±3.9	76.7 ±5.2	24.7	50.4	48.4
forge	C	47.4 ±4.2	40.6 ±6.2	48.9 ±4.3	47.5 ±5.4	22.6	86.0	30.7
furnish	C	66.0 ±6.9	60.0 ±6.5	66.7 ±5.9	68.7 ±5.0	23.0	52.7	40.9
hail	C	80.7 ±2.9	82.4 ±4.3	82.3 ±2.4	80.4 ±2.2	14.9	22.2	45.8
halt	C	85.6 ±3.0	84.0 ±2.4	85.6 ±2.8	86.0 ±3.1	2.0	2.4	12.2
part	C	72.4 ±3.9	72.0 ±5.9	75.3 ±3.8	75.3 ±6.2	32.2	74.8	56.6
plough	C	64.8 ±6.0	52.9 ±6.5	68.1 ±4.5	68.4 ±4.6	35.5	109.3	52.7
plug	C	56.0 ±5.6	49.7 ±6.4	53.0 ±3.6	53.7 ±3.8	21.7	69.3	31.6
pour	C	48.6 ±5.5	48.3 ±5.5	53.0 ±5.7	50.3 ±5.8	28.6	117.2	37.9
say	A	88.0 ±0.7	88.8 ±1.8	89.6 ±2.1	89.4 ±2.2	4.4	5.2	29.8
smash	C	61.7 ±2.2	62.7 ±2.7	64.4 ±3.1	64.8 ±5.9	11.0	20.5	23.5
smell	C	57.3 ±3.1	54.3 ±4.5	59.3 ±4.5	56.7 ±1.8	23.0	63.4	36.1
steer	C	42.4 ±5.3	37.7 ±4.6	43.0 ±6.9	42.4 ±5.2	25.4	143.7	30.8
submit	B	87.6 ±4.0	83.6 ±5.7	86.0 ±2.5	84.8 ±5.5	15.3	21.6	52.2
swell	C	38.7 ±5.1	40.7 ±4.1	45.7 ±2.9	39.0 ±5.5	28.7	169.1	34.6
tell	A	72.8 ±2.7	71.2 ±4.2	76.4 ±1.2	74.8 ±2.6	11.2	17.2	32.2
throw	B	42.6 ±3.5	36.6 ±4.5	43.3 ±3.1	46.5 ±2.9	20.6	90.7	26.6
trouble	C	66.7 ±5.0	63.0 ±5.1	68.4 ±4.3	68.3 ±4.3	24.0	54.0	43.1
wake	C	75.4 ±4.6	69.7 ±4.3	76.0 ±2.5	77.4 ±4.2	31.0	68.9	56.3
yield	B	41.7 ±7.6	37.7 ±5.6	44.7 ±5.2	40.0 ±6.7	16.3	57.6	22.8
A		84.6 ±1.2	84.9 ±2.3	86.7 ±2.1	86.2 ±2.3	6.6	8.8	31.8
B		60.6 ±5.7	59.7 ±5.1	61.4 ±3.8	61.4 ±3.6	11.4	33.4	22.0
C		64.1 ±4.4	61.7 ±4.5	66.0 ±4.0	65.5 ±4.4	22.3	67.3	38.3
Celkom		80.2 ±2.0	80.2 ±2.8	82.1 ±2.5	81.7 ±2.6	8.3	16.0	31.1

Tabuľka D.19: Výsledky experimentu **Default-FS+NER**. Modely klasifikátorov boli natrénované pomocou všetkých morfo-syntaktických rysov a troch rysov z rozpoznávača menných entít. Najlepším algoritmom je SVM, hoci medzi výsledkami algoritmov neexistuje štatisticky signifikantný rozdiel.

D.20 Experiment Best58+MU44

						Zlepšenie		
Sloveso	**S.**	**DT**	**kNN**	**SVM**	**ADA**	**r**	**%**	**ERD**
access	C	75.7 ±4.2	71.0 ±4.8	77.0 ±4.0	76.3 ±4.4	30.0	63.9	56.6
ally	C	64.1 ±5.7	62.8 ±7.0	66.4 ±4.2	66.4 ±3.0	18.7	39.2	35.8
arrive	B	71.9 ±5.8	67.6 ±5.6	75.6 ±6.1	72.8 ±3.1	7.6	11.2	23.8
breathe	C	70.9 ±4.2	64.8 ±3.3	72.3 ±4.4	71.7 ±4.8	34.5	91.5	55.5
claim	A	82.6 ±3.2	79.4 ±2.8	83.4 ±3.1	83.0 ±2.7	15.6	23.0	48.4
cool	C	57.7 ±6.4	54.4 ±5.6	64.0 ±5.1	63.7 ±8.1	36.8	134.9	50.6
crush	C	49.5 ±6.5	42.3 ±4.1	49.7 ±8.2	47.7 ±2.5	20.3	69.0	28.8
cry	B	66.4 ±6.1	68.5 ±4.2	74.8 ±4.5	75.6 ±3.5	22.5	42.9	47.1
deny	B	53.7 ±6.2	48.4 ±5.4	57.7 ±2.1	56.0 ±3.9	13.3	30.1	23.9
enlarge	C	76.7 ±1.7	77.3 ±3.0	80.0 ±2.5	82.4 ±3.6	3.4	4.4	14.5
enlist	C	85.7 ±6.0	79.7 ±6.5	85.4 ±4.5	85.0 ±5.2	36.4	74.2	71.3
forge	C	50.0 ±4.2	42.0 ±4.3	52.9 ±5.6	52.0 ±3.0	26.6	101.1	36.1
furnish	C	69.7 ±5.0	59.0 ±4.9	68.0 ±5.7	66.7 ±5.8	24.4	55.8	43.2
hail	C	80.7 ±2.1	80.0 ±4.5	83.0 ±4.7	82.0 ±2.7	15.6	23.2	48.0
halt	C	84.4 ±1.9	81.2 ±3.8	86.4 ±2.7	85.2 ±5.3	2.8	3.3	17.0
part	C	70.4 ±4.2	70.9 ±5.8	74.3 ±4.8	74.3 ±4.4	31.2	72.4	54.8
plough	C	65.3 ±5.3	60.1 ±5.9	67.7 ±5.0	69.6 ±4.2	35.1	108.1	52.1
plug	C	57.3 ±8.6	49.7 ±5.0	60.3 ±5.8	58.0 ±4.4	29.0	92.6	42.2
pour	C	42.7 ±3.3	45.0 ±4.8	54.4 ±2.6	47.3 ±8.1	29.9	122.6	39.6
say	A	88.8 ±2.1	88.4 ±4.6	90.2 ±1.9	89.4 ±2.4	5.0	5.9	33.8
smash	C	68.0 ±5.8	64.4 ±7.3	67.7 ±4.1	69.0 ±5.0	14.3	26.7	30.6
smell	C	54.7 ±3.2	49.7 ±5.2	60.7 ±5.6	56.4 ±3.5	24.4	67.2	38.3
steer	C	42.7 ±4.7	39.6 ±5.7	44.7 ±4.3	42.4 ±4.9	27.0	152.9	32.8
submit	B	87.6 ±4.0	82.0 ±4.7	86.4 ±3.6	84.8 ±4.3	15.7	22.1	53.5
swell	C	39.7 ±4.7	42.3 ±5.9	49.2 ±8.4	42.0 ±6.1	32.3	190.2	38.9
tell	A	78.0 ±1.2	73.0 ±3.3	80.6 ±2.5	77.4 ±2.3	15.4	23.6	44.3
throw	B	46.6 ±4.7	37.3 ±3.9	49.7 ±2.1	45.0 ±4.2	27.0	118.9	34.9
trouble	C	66.0 ±4.7	62.0 ±4.3	69.0 ±4.2	68.3 ±4.3	24.6	55.5	44.3
wake	C	75.4 ±4.6	69.3 ±4.3	77.0 ±3.6	76.7 ±3.9	32.0	71.2	58.2
yield	B	50.4 ±4.7	43.3 ±8.2	52.4 ±3.9	52.4 ±7.8	24.0	84.7	33.5
A		86.4 ±2.0	84.9 ±4.2	87.9 ±2.1	86.8 ±2.5	7.8	10.5	37.0
B		62.5 ±5.4	57.0 ±5.2	65.7 ±3.8	63.4 ±4.0	15.8	45.0	31.5
C		65.2 ±4.3	61.6 ±4.8	68.4 ±4.5	67.0 ±4.7	24.7	74.6	42.1
Celkom		82.0 ±2.6	79.9 ±4.4	83.8 ±2.5	82.5 ±2.8	10.0	19.3	36.7

Tabuľka D.20: Výsledky experimentu **Best58+MU44**. Modely klasifikátorov boli natrénované pomocou 58 morfo-syntaktických rysov a 44 rysov z množiny MU44. Najlepším algoritmom je SVM, hoci medzi výsledkami algoritmov neexistuje štatisticky signifikantný rozdiel.

D.21 Experiment Best58+AU124

						Zlepšenie		
Sloveso	**S.**	**DT**	**kNN**	**SVM**	**ADA**	**r**	**%**	**ERD**
access	C	76.7 ±2.8	75.7 ±3.7	77.7 ±3.4	76.3 ±4.3	30.7	65.3	57.9
ally	C	63.2 ±3.4	62.4 ±6.3	65.6 ±5.9	66.4 ±2.7	17.9	37.6	34.3
arrive	B	72.4 ±5.7	69.2 ±4.6	75.6 ±6.5	72.8 ±3.1	7.6	11.2	23.7
breathe	C	69.1 ±3.8	68.3 ±3.8	72.9 ±3.7	71.7 ±4.8	35.1	93.0	56.4
claim	A	83.0 ±3.5	81.2 ±3.1	84.2 ±3.3	83.0 ±2.9	16.4	24.2	50.9
cool	C	59.7 ±7.6	62.0 ±7.0	63.7 ±4.2	63.7 ±8.1	36.4	133.7	50.1
crush	C	48.6 ±5.1	43.7 ±6.8	50.6 ±6.0	47.7 ±2.9	21.2	71.9	30.0
cry	B	66.4 ±6.1	68.8 ±4.8	74.0 ±7.2	75.6 ±3.7	21.6	41.4	45.4
deny	B	55.0 ±5.5	58.7 ±4.2	57.7 ±2.4	56.0 ±4.0	13.4	30.1	24.0
enlarge	C	76.7 ±1.7	80.0 ±2.3	79.4 ±2.3	82.4 ±3.9	2.7	3.5	11.6
enlist	C	82.4 ±4.4	82.1 ±5.8	86.4 ±4.6	85.0 ±5.1	37.4	76.3	73.2
forge	C	52.0 ±3.0	46.0 ±5.4	53.4 ±4.1	52.0 ±3.4	27.1	103.2	36.8
furnish	C	70.7 ±5.6	67.0 ±6.4	70.7 ±5.4	66.7 ±5.8	27.0	61.8	47.9
hail	C	81.7 ±2.1	80.7 ±2.8	83.7 ±4.3	82.0 ±2.7	16.3	24.2	50.0
halt	C	84.4 ±1.9	85.6 ±2.0	86.0 ±2.7	85.2 ±5.3	2.0	2.4	12.3
part	C	70.4 ±4.2	69.9 ±5.5	73.3 ±4.9	74.3 ±4.4	30.2	70.1	53.1
plough	C	66.5 ±5.2	60.9 ±4.7	68.0 ±3.6	69.6 ±4.2	35.5	109.3	52.6
plug	C	57.3 ±9.3	51.0 ±6.2	59.0 ±6.9	58.0 ±4.4	27.7	88.6	40.3
pour	C	42.7 ±5.2	47.4 ±7.4	55.0 ±4.8	47.3 ±8.4	30.6	125.4	40.5
say	A	88.8 ±2.1	89.0 ±1.7	90.0 ±2.0	89.4 ±2.4	4.8	5.7	32.5
smash	C	68.0 ±6.7	66.1 ±4.6	69.7 ±3.6	69.0 ±4.9	16.3	30.4	34.9
smell	C	56.7 ±4.0	52.7 ±6.1	58.7 ±5.5	56.4 ±3.7	22.4	61.6	35.1
steer	C	42.4 ±4.8	43.7 ±4.6	42.6 ±6.6	42.4 ±4.9	25.0	141.4	30.3
submit	B	87.6 ±4.0	83.6 ±4.1	85.2 ±4.6	84.8 ±4.4	14.5	20.4	49.4
swell	C	42.3 ±6.9	45.0 ±6.0	52.3 ±7.4	42.0 ±6.3	35.3	208.0	42.5
tell	A	77.4 ±1.3	72.4 ±2.1	79.6 ±2.4	77.4 ±2.4	14.4	22.1	41.4
throw	B	45.7 ±3.1	40.4 ±3.5	48.3 ±4.1	45.0 ±4.2	25.6	112.8	33.1
trouble	C	67.3 ±6.2	65.0 ±5.4	67.0 ±4.1	68.3 ±4.3	22.6	51.0	40.7
wake	C	75.4 ±4.6	71.3 ±2.1	76.7 ±4.0	76.7 ±3.9	31.7	70.4	57.6
yield	B	52.7 ±6.3	47.3 ±7.3	53.3 ±5.0	52.4 ±7.8	25.0	88.0	34.8
A		86.3 ±2.1	85.5 ±1.9	87.7 ±2.2	86.8 ±2.5	7.5	10.2	35.8
B		62.9 ±5.0	61.1 ±4.5	65.3 ±4.8	63.4 ±4.1	15.4	43.8	30.7
C		65.4 ±4.4	64.5 ±4.7	68.5 ±4.4	67.0 ±4.8	24.8	75.1	42.1
Celkom		82.0 ±2.6	81.0 ±2.4	83.6 ±2.7	82.5 ±2.8	9.7	18.9	35.6

Tabuľka D.21: Výsledky experimentu **Best58+AU124**. Modely klasifikátorov boli natrénované pomocou 58 morfo-syntaktických rysov a 124 rysov z množiny AU124. Najlepším algoritmom je SVM, hoci medzi výsledkami algoritmov neexistuje štatisticky signifikantný rozdiel.

D.22 Najlepšie rysy z Best58 a MU44 podľa hladového algoritmu

Sloveso	Najúspešnejšie rysy
access	X1n_to, X1p_nominal, any_obj,
ally	X1n_nominal, **subj_H**, tense_vbg, tense_vb, X1n_to,
arrive	X1p_to, prep_at, X1p_verbs, advmod, X1p_nominal,
breathe	X1p_adverbial, prep_into, **obj_cm**, **obj_percep**, n_subj, obj_pl, tense_vb, tense_vbg, **obj_ac**, prt_none
claim	X1n_to, **obj_unit**, dobj, X1n_nominal, **subj_f**, tense_vb, any_obj,
cool	any_obj, n_subj, X1p_to, dobj, tense_vbn, nsubjpass, X1p_verbs,
crush	**obj_act**, X1p_to, **obj_cm**, tense_vbg, dobj, tense_vbn, X1p_adverbial, X1n_nominal
cry	prt_none, prep_for, ccomp, X1n_nominal, X2n_nominal,
deny	X1n_verbs, prep_to, **obj_cc**, prep_on, X1n_adjective,
enlarge	prep_on,
enlist	any_obj, **obj_act**, X1p_be, complm, X1n_nominal,
forge	prep_with, **obj_cc**, **obj_tool**, complm, X1p_to, **obj_sem**, **obj_f**, X2n_adverbial
furnish	prep_with, X2p_be, tense_vbn, X1p_adverbial, X1p_verbs,
hail	prep_from, **obj_V**
halt	**subj_A**
part	X1n_nominal, prep_from, X1n_adverbial, tense_vbn, advmod, X2n_nominal, tense_vbd,
plough	any_obj, prep_into, prt_none, tense_vb, prep_through, X1n_nominal, X1p_be,
plug	**obj_L**, **obj_ac**, **subj_cc**, X2n_adjective, **subj_tool**, X1n_adverbial, any_obj, tense_vb, X2p_nominal, prep_into, X3p_verbs, tense_vbg
pour	any_obj, prt_out, prep_from, X1n_nominal, prt_away,
say	X1n_to, modality_2
smash	prep_into, **obj_H**, X1n_nominal,
smell	X1n_nominal, prep_none, n_subj, X3n_nominal, **obj_A**,
steer	any_obj, X1n_adjective, X2n_adjective, prep_away_from, **obj_V**, tense_vbg, X2n_adverbial,
submit	ccomp, X1n_adverbial
swell	obj_pl, tense_vbn, tense_vbg, prep_to, **obj_unit**, **subj_H**, X1p_verbs,
tell	X2n_to, prep_about, prep_of, **obj_sem**, X1n_to, ccomp
throw	**obj_percep**, prep_at, prep_into, prt_down, prep_behind, prep_out_of, prt_up, prt_out, X2n_adverbial,
trouble	X1n_to, prep_with
wake	tense_vbn, X1n_nominal, tense_vbg, X2n_to
yield	**obj_sem**, any_obj, tense_vb, ccomp, X2p_adjective, **obj_unit**, tense_vbn,

Tabuľka D.22: Najúspešnejšie rysy vybrané hladovým algoritmom z množín Best58 a MU44 individuálne pre každé sloveso. Hladový algoritmus skončil s pridávaním rysov po dosiahnutí prvého (lokálneho) maxima v Accuracy.

D.23 Najlepšie rysy z Best58 a AU124 podľa hladového algoritmu

Sloveso	Najúspešnejšie rysy
access	X1n_to, X1p_nominal, any_obj,
ally	X1n_nominal, **subj_H**, tense_vbg, tense_vb, X1n_to,
arrive	**subj_per**, X1p_to
breathe	X 1p_adverbial, prep_into, **obj_cm**, **obj_sem**, n_subj, obj_pl, tense_vb, tense_vbg, **obj_per**, **obj_ac**, prt_none,
claim	X1n_to, **obj_mon**, dobj, X1n_nominal, tense_vb, any_obj, **obj_ac**, X2n_nominal
cool	any_obj, n_subj, X1p_to, dobj, tense_vbn, nsubjpass, X1p_verbs,
crush	**obj_act**, X1p_to, **obj_mat**, tense_vbg, X1n_nominal, tense_vbn, X1p_adverbial, **obj_f**
cry	prt_none, prep_for, ccomp, X1n_nominal, X2n_nominal,
deny	X1n_verbs, prep_to, **obj_cc**, **obj_sit**, prep_on, X1n_adjective
enlarge	prep_on,
enlist	any_obj, **obj_ac**, X1p_be, complm, X1n_nominal,
forge	prep_with, **obj_tool**, **obj_cc**, complm, X1p_to, **obj_sem**, X1n_adverbial, tense_vbn, any_obj,
furnish	prep_with, **obj_build**, X2p_be, dobj, advmod, tense_vbg, tense_vb, tense_vbn, prep_none, X1p_verbs, **obj_ac**,
hail	prep_from, **obj_V**
halt	**subj_A**,
part	X1n_nominal, prep_from, X1n_adverbial, tense_vbn, advmod, X2n_nominal, tense_vbd,
plough	any_obj, prep_into, prt_none, tense_vb, prep_through, X1n_nominal, X1p_be,
plug	**obj_L**, **obj_geom**, X2n_adjective, prep_into, any_obj, X1n_adverbial, tense_vb, X2n_nominal, X3p_verbs,
pour	any_obj, prt_out, prep_from, X1n_nominal, prt_away,
say	X1n_to, modality_2
smash	prep_into, **obj_H**, X1n_nominal,
smell	X1n_nominal, prep_none, n_subj, X3n_nominal, **obj_mat**, **obj_A**
steer	any_obj, X1n_adjective, X2n_adjective, prep_away_from, **obj_V**, tense_vbg, X2n_adverbial,
submit	ccomp, X1n_adverbial
swell	obj_pl, tense_vbn, tense_vbg, prep_to, **obj_mon**, X1p_verbs
tell	X2n_to, prep_about, prep_of, **obj_sem**, X1n_to, ccomp
throw	**obj_percep**, prep_at, prep_into, prt_down, prep_behind, prep_out_of, prt_up, prt_out, X2n_adverbial,
trouble	X1n_to, prep_with, **subj_H**, X1n_nominal, tense_vbn, n_subj, X1p_adverbial, X1n_adverbial
wake	tense_vbn, X1n_nominal, tense_vbg, X2n_to
yield	**obj_sem**, any_obj, tense_vb, ccomp, X2p_adjective, **obj_mon**, tense_vbn,

Tabuľka D.23: Najúspešnejšie rysy vybrané hladovým algoritmom z množín Best58 a AU124 individuálne pre každé sloveso. Hladový algoritmus skončil s pridávaním rysov po dosiahnutí prvého (lokálneho) maxima v Accuracy.

D.24 Experiment BestMU

						Zlepšenie		
Sloveso	**S.**	**DT**	**kNN**	**SVM**	**ADA**	**r**	**%**	**ERD**
access	C	75.7 ±4.2	76.0 ±4.0	77.3 ±4.5	76.7 ±3.1	29.7	63.1	56.0
ally	C	63.3 ±5.8	64.0 ±3.9	64.8 ±5.9	66.4 ±2.7	17.1	35.9	32.7
arrive	B	70.8 ±3.9	71.6 ±4.6	72.4 ±5.4	71.2 ±2.0	4.4	6.5	13.9
breathe	C	70.9 ±4.2	65.1 ±2.8	70.9 ±6.1	71.7 ±4.1	33.1	87.7	53.2
claim	A	82.8 ±3.0	81.4 ±2.9	83.6 ±3.0	83.6 ±2.4	15.8	23.3	49.1
cool	C	58.7 ±7.6	61.0 ±6.9	65.0 ±6.5	63.7 ±8.7	37.7	138.4	51.9
crush	C	50.3 ±6.2	45.4 ±7.3	51.5 ±8.4	54.9 ±6.6	22.0	74.9	31.2
cry	B	66.8 ±6.9	67.6 ±4.3	72.8 ±2.9	75.6 ±3.4	20.5	39.1	42.9
deny	B	52.7 ±6.1	59.0 ±3.8	59.0 ±4.6	56.6 ±5.0	14.7	33.1	26.4
enlarge	C	78.7 ±1.7	80.0 ±4.0	80.4 ±3.5	83.4 ±2.7	3.7	4.9	16.0
enlist	C	83.7 ±5.4	78.4 ±4.5	83.1 ±3.7	85.0 ±2.7	34.1	69.5	66.8
forge	C	50.6 ±3.2	45.7 ±4.4	53.7 ±6.2	52.8 ±4.1	27.4	104.4	37.2
furnish	C	69.3 ±4.3	66.7 ±3.2	70.3 ±5.3	67.4 ±5.0	26.7	61.1	47.4
hail	C	80.7 ±2.1	83.4 ±4.4	82.4 ±3.0	82.0 ±3.9	15.0	22.2	45.9
halt	C	87.2 ±3.9	85.2 ±1.6	86.8 ±3.4	86.8 ±3.8	3.2	3.8	19.5
part	C	70.0 ±5.0	69.6 ±4.3	73.3 ±2.4	75.3 ±4.7	30.2	70.1	53.1
plough	C	65.3 ±5.3	62.5 ±6.0	69.6 ±3.5	69.6 ±3.8	37.1	114.2	55.0
plug	C	55.0 ±7.3	52.0 ±5.7	56.7 ±4.2	59.0 ±4.6	25.4	81.1	36.9
pour	C	42.7 ±3.3	48.7 ±5.2	54.6 ±4.6	52.0 ±7.7	30.2	123.7	40.0
say	A	88.8 ±2.3	88.4 ±1.6	90.2 ±2.6	89.4 ±2.2	5.0	5.9	33.9
smash	C	68.0 ±5.8	67.7 ±5.0	69.7 ±5.4	69.0 ±5.6	16.3	30.5	34.9
smell	C	54.7 ±3.2	52.3 ±5.5	59.4 ±5.1	59.7 ±4.8	23.1	63.5	36.2
steer	C	42.1 ±4.9	40.7 ±4.7	45.3 ±3.3	43.7 ±5.9	27.7	156.7	33.6
submit	B	87.6 ±3.7	86.0 ±3.5	84.8 ±3.3	85.6 ±4.8	14.1	19.9	48.1
swell	C	40.3 ±5.2	43.3 ±5.9	51.0 ±8.8	48.3 ±6.0	34.0	200.2	40.9
tell	A	76.8 ±3.7	75.0 ±4.3	78.8 ±2.6	79.2 ±3.7	13.6	20.9	39.1
throw	B	46.6 ±4.7	43.5 ±2.9	47.5 ±2.3	49.1 ±3.1	24.8	109.3	32.1
trouble	C	66.0 ±5.1	64.7 ±4.4	69.0 ±3.4	68.0 ±5.0	24.6	55.5	44.3
wake	C	75.4 ±4.6	73.6 ±5.9	77.7 ±3.5	77.0 ±4.5	32.7	72.6	59.4
yield	B	53.4 ±3.5	47.3 ±5.7	54.0 ±5.2	52.4 ±4.6	25.7	90.5	35.8
A		86.2 ±2.6	85.5 ±2.2	87.7 ±2.6	87.1 ±2.5	7.5	10.1	36.3
B		62.1 ±4.7	62.8 ±4.0	64.4 ±4.2	63.9 ±3.6	14.5	42.4	27.8
C		65.4 ±4.4	64.6 ±4.6	68.5 ±4.7	68.4 ±4.8	24.8	75.2	42.4
Celkom		81.8 ±3.0	81.2 ±2.6	83.5 ±3.0	83.0 ±2.8	9.6	18.7	35.7

Tabuľka D.24: Výsledky experimentu **BestMU**. Modely klasifikátorov boli natrénované pomocou 69 rysov z množiny BestMU. Najlepším algoritmom je SVM, hoci medzi výsledkami algoritmov neexistuje štatisticky signifikantný rozdiel.

D.25 Experiment BestAU

						Zlepšenie		
Sloveso	**S.**	**DT**	**kNN**	**SVM**	**ADA**	**r**	**%**	**ERD**
access	C	76.7 ±2.8	75.0 ±5.2	78.0 ±4.0	77.7 ±2.7	30.7	65.3	57.9
ally	C	62.8 ±3.7	66.0 ±1.9	66.8 ±4.6	66.4 ±4.6	19.1	40.1	36.6
arrive	B	71.2 ±6.0	70.0 ±3.1	72.8 ±5.2	72.4 ±3.8	4.9	7.1	15.1
breathe	C	69.1 ±3.7	66.6 ±3.5	72.3 ±4.3	71.7 ±3.1	34.5	91.5	55.5
claim	A	83.2 ±3.5	81.6 ±3.7	83.0 ±4.2	83.2 ±2.8	15.2	22.4	47.2
cool	C	60.0 ±6.7	62.0 ±6.9	63.4 ±4.2	65.0 ±7.0	36.1	132.5	49.7
crush	C	49.4 ±5.1	45.1 ±5.1	51.4 ±5.8	48.3 ±5.6	22.0	74.8	31.2
cry	B	66.8 ±6.9	70.4 ±3.7	74.4 ±2.7	75.6 ±4.1	22.1	42.2	46.3
deny	B	55.0 ±5.5	59.0 ±2.9	61.3 ±4.7	56.3 ±6.6	17.0	38.4	30.6
enlarge	C	78.7 ±1.7	80.0 ±3.1	80.7 ±3.5	82.0 ±3.2	4.1	5.3	17.4
enlist	C	81.8 ±7.1	78.4 ±4.6	83.4 ±2.8	85.0 ±2.6	34.4	70.1	67.4
forge	C	52.3 ±3.4	46.3 ±4.7	52.3 ±5.4	53.7 ±4.0	26.0	98.9	35.3
furnish	C	70.3 ±6.1	66.0 ±6.2	69.7 ±5.5	69.3 ±5.6	26.0	59.6	46.2
hail	C	82.7 ±2.2	82.7 ±3.8	83.4 ±4.5	82.0 ±3.9	16.0	23.7	49.0
halt	C	86.4 ±3.6	84.4 ±2.7	86.8 ±2.4	86.4 ±4.1	3.2	3.8	19.5
part	C	71.7 ±3.6	70.0 ±5.4	72.0 ±2.1	75.6 ±5.6	28.9	67.1	50.8
plough	C	67.6 ±4.0	63.3 ±4.6	70.8 ±3.0	69.7 ±4.2	38.3	117.9	56.8
plug	C	56.3 ±7.6	51.7 ±4.9	59.0 ±4.1	58.0 ±5.9	27.7	88.5	40.3
pour	C	42.7 ±5.2	50.4 ±5.0	56.0 ±4.7	53.0 ±6.5	31.6	129.4	41.8
say	A	89.0 ±2.2	88.4 ±1.4	90.6 ±2.3	89.6 ±2.4	5.4	6.4	36.6
smash	C	68.0 ±7.0	64.7 ±3.1	69.7 ±4.3	69.6 ±6.4	16.3	30.5	34.9
smell	C	57.3 ±2.3	56.0 ±2.4	62.0 ±4.2	59.7 ±3.9	25.7	70.8	40.4
steer	C	42.7 ±4.7	40.0 ±5.2	43.7 ±7.2	43.0 ±3.1	26.0	147.2	31.6
submit	B	87.6 ±3.7	85.2 ±3.8	85.2 ±2.9	85.6 ±3.9	14.5	20.4	49.4
swell	C	40.4 ±5.8	45.0 ±4.2	50.3 ±6.5	42.3 ±7.7	33.3	196.5	40.2
tell	A	77.0 ±3.1	76.6 ±2.7	78.4 ±2.0	79.0 ±3.1	13.2	20.3	38.0
throw	B	37.2 ±2.8	36.7 ±3.9	42.0 ±2.2	44.0 ±2.4	19.3	85.0	25.0
trouble	C	66.7 ±4.9	63.0 ±6.2	69.0 ±4.0	69.0 ±5.0	24.7	55.6	44.3
wake	C	75.4 ±4.6	74.0 ±3.1	77.7 ±3.9	78.0 ±4.1	32.7	72.6	59.4
yield	B	52.0 ±3.8	46.0 ±6.8	52.7 ±5.0	53.0 ±5.4	24.3	85.7	33.9
A		86.4 ±2.5	85.8 ±1.9	87.8 ±2.4	87.2 ±2.5	7.7	10.2	37.8
B		60.9 ±4.9	60.9 ±3.6	64.1 ±4.1	63.3 ±4.4	14.2	39.2	28.1
C		65.8 ±4.4	64.8 ±4.2	68.8 ±4.3	68.2 ±4.6	25.1	75.8	43.1
Celkom		81.8 ±2.9	81.2 ±2.2	83.6 ±2.7	82.9 ±2.9	9.7	18.4	37.0

Tabuľka D.25: Výsledky experimentu **BestAU**. Modely klasifikátorov boli natrénované pomocou 65 rysov z množiny BestAU. Najlepším algoritmom je SVM, hoci medzi výsledkami algoritmov neexistuje štatisticky signifikantný rozdiel.

D.26 Experiment GreedyAU

						Zlepšenie		
Sloveso	**S.**	**DT**	**kNN**	**SVM**	**ADA**	**r**	**%**	**ERD**
access	C	76.7 ±2.8	77.0 ±4.7	77.3 ±3.9	77.7 ±2.7	30.3	64.5	57.2
ally	C	63.6 ±3.0	64.8 ±3.9	63.9 ±6.5	66.4 ±1.9	16.2	34.1	31.0
arrive	B	70.8 ±4.9	72.4 ±5.0	72.0 ±4.3	70.8 ±3.5	4.0	5.9	12.6
breathe	C	69.1 ±3.7	66.0 ±3.5	70.9 ±4.7	71.7 ±4.7	33.1	87.7	53.2
claim	A	83.2 ±3.5	81.4 ±3.6	82.8 ±3.3	83.0 ±3.2	15.0	22.1	46.6
cool	C	60.3 ±6.3	60.0 ±7.4	65.0 ±7.0	63.7 ±7.1	37.7	138.4	51.9
crush	C	48.9 ±3.9	46.5 ±5.8	50.6 ±7.0	50.3 ±5.2	21.2	71.9	30.0
cry	B	66.8 ±6.9	72.4 ±3.7	74.8 ±3.6	76.8 ±2.0	22.5	43.0	47.2
deny	B	55.0 ±5.5	58.4 ±3.5	59.7 ±4.3	58.0 ±5.7	15.4	34.6	27.6
enlarge	C	78.7 ±1.7	79.7 ±4.3	80.4 ±4.2	82.4 ±3.4	3.7	4.9	16.0
enlist	C	82.4 ±4.4	80.7 ±4.2	85.4 ±3.2	85.0 ±3.1	36.4	74.3	71.3
forge	C	52.0 ±3.0	44.6 ±4.7	51.7 ±4.4	53.4 ±3.7	25.4	96.8	34.5
furnish	C	70.0 ±4.8	66.3 ±6.2	70.3 ±5.0	69.4 ±4.0	26.7	61.1	47.3
hail	C	82.0 ±1.9	82.0 ±3.5	84.0 ±2.3	82.3 ±2.9	16.6	24.7	51.0
halt	C	86.4 ±3.8	86.0 ±2.3	88.8 ±3.2	87.6 ±4.1	5.2	6.2	31.6
part	C	70.0 ±5.0	71.0 ±5.5	72.3 ±2.5	73.7 ±6.4	29.2	67.8	51.3
plough	C	66.0 ±4.6	63.7 ±5.0	70.0 ±2.5	68.5 ±3.6	37.5	115.4	55.6
plug	C	55.3 ±8.6	51.0 ±5.3	56.0 ±1.9	58.7 ±5.9	24.7	78.9	36.0
pour	C	47.7 ±6.6	48.0 ±6.3	54.3 ±4.7	48.7 ±5.9	29.9	122.4	39.6
say	A	88.8 ±2.3	88.6 ±1.6	90.0 ±2.6	89.4 ±2.3	4.8	5.7	32.6
smash	C	67.7 ±6.9	66.8 ±5.3	69.0 ±3.9	71.3 ±5.8	15.6	29.2	33.4
smell	C	57.3 ±3.6	54.7 ±6.9	59.4 ±4.9	59.3 ±4.0	23.1	63.5	36.2
steer	C	42.7 ±4.7	39.0 ±3.5	44.4 ±4.4	46.7 ±5.6	26.7	151.2	32.4
submit	B	87.6 ±3.7	84.8 ±3.9	85.2 ±2.6	84.8 ±5.1	14.5	20.4	49.4
swell	C	40.4 ±5.8	45.3 ±7.2	51.0 ±5.1	47.0 ±6.3	34.0	200.4	41.0
tell	A	77.0 ±2.4	75.4 ±2.5	79.4 ±2.2	78.0 ±3.4	14.2	21.8	40.8
throw	B	45.3 ±3.5	42.8 ±3.0	48.1 ±3.4	47.3 ±3.5	25.4	111.9	32.9
trouble	C	65.7 ±4.9	63.6 ±2.9	67.7 ±3.6	68.3 ±4.3	23.3	52.6	41.9
wake	C	75.4 ±4.6	71.9 ±4.7	77.7 ±3.1	78.0 ±4.6	32.7	72.6	59.4
yield	B	54.3 ±5.0	48.7 ±5.4	54.0 ±4.6	53.0 ±7.2	25.6	90.4	35.8
A		86.3 ±2.4	85.7 ±1.9	87.5 ±2.6	86.9 ±2.5	7.4	10.0	35.3
B		62.5 ±4.8	63.0 ±4.1	64.8 ±3.9	63.8 ±4.4	14.8	43.4	28.3
C		65.9 ±4.4	64.5 ±4.8	68.5 ±4.2	68.3 ±4.6	24.8	74.7	43.3
Celkom		81.9 ±2.8	81.4 ±2.4	83.4 ±2.8	82.7 ±2.9	9.5	18.6	35.0

Tabuľka D.26: Výsledky experimentu **GreedyAU**. Modely klasifikátorov boli natrénované pomocou 77 rysov z množiny GreedyAU. Najlepším algoritmom je SVM, hoci medzi výsledkami algoritmov neexistuje štatisticky signifikantný rozdiel.

D.27 Evaluácia Best58 na testovacích údajoch

Sloveso	S.	Kernel	D.	Gamma	C.	Priemer	Train	Test	MULTI
access	C	polynomial	1	0.0028	1	78.0 ±3.0	78.3	78.0	90.0
ally	C	polynomial	1	0.09	1	66.0 ±4.7	72.8	56.0	56.0
arrive	B	polynomial	1	0.023	1	70.8 ±4.2	74.0	62.0	66.0
breathe	C	polynomial	1	0.027	1	66.6 ±7.0	75.4	68.0	72.0
claim	A	polynomial	2	0.026	1	81.6 ±2.4	85.6	84.0	86.0
cool	C	polynomial	1	0.01	1	63.0 ±7.6	82.7	42.0	50.0
crush	C	polynomial	1	0.03	1	42.3 ±3.7	54.6	34.0	38.0
cry	B	polynomial	1	0.06	1	72.4 ±4.5	93.6	70.0	76.0
deny	B	polynomial	1	0.04	1	56.7 ±2.2	64.3	46.0	50.0
enlarge	C	polynomial	1	0.07	1	82.0 ±3.5	99.0	64.0	74.0
enlist	C	polynomial	1	0.024	1	74.4 ±4.3	82.0	61.7	66.0
forge	C	polynomial	1	0.06	1	48.6 ±3.8	58.0	44.0	50.0
furnish	C	polynomial	1	0.024	1	66.0 ±5.3	84.0	62.0	66.0
hail	C	polynomial	1	0.04	1	84.7 ±3.5	88.7	90.0	90.0
halt	C	polynomial	1	0.08	1	85.2 ±2.3	86.8	84.0	86.0
part	C	polynomial	1	0.06	1	74.0 ±3.5	91.7	56.0	64.0
plough	C	polynomial	1	0.05	1	68.9 ±5.6	82.8	56.0	58.0
plug	C	polynomial	2	0.04	1	50.3 ±3.1	73.3	43.5	47.8
pour	C	polynomial	1	0.07	1	50.7 ±8.5	96.0	46.0	52.0
say	A	polynomial	1	0.04	1	90.2 ±2.6	92.6	86.0	86.0
smash	C	polynomial	1	0.08	1	64.4 ±7.7	82.0	64.0	70.0
smell	C	polynomial	1	0.09	1	57.4 ±5.1	67.7	62.0	68.0
steer	C	polynomial	1	0.04	1	40.0 ±3.2	56.7	46.0	52.0
submit	B	polynomial	2	0.026	1	85.6 ±5.8	90.0	88.0	90.0
swell	C	polynomial	1	0.024	1	47.0 ±7.8	83.3	42.0	44.0
tell	A	polynomial	1	0.04	1	76.0 ±1.8	87.2	72.0	72.0
throw	B	polynomial	1	0.022	1	43.2 ±3.2	50.3	42.0	46.0
trouble	C	polynomial	1	0.04	1	69.7 ±3.2	72.7	72.0	76.0
wake	C	polynomial	1	0.09	1	76.3 ±2.5	86.7	80.0	90.0
yield	B	polynomial	1	0.028	1	46.7 ±6.1	66.3	44.0	44.0
A						87.0 ±2.4	91.0	83.5	83.7
B						62.1 ±3.8	69.6	56.4	60.0
C						66.0 ±4.8	80.4	62.0	67.3
Celkom						82.4 ±2.8	87.6	78.6	79.6

Tabuľka D.27: Výsledky modelu **Best58** na testovacích údajoch. Modely klasifikátorov boli natrénované pomocou 58 rysov z množiny Best58, pomocou algoritmu SVM.

D.28 Evaluácia BestAU na testovacích údajoch

Sloveso	S.	Kernel	D.	Gamma	C.	Priemer	Train	Test	MULTI
access	C	polynomial	1	0.0028	1	78.0 ±4.0	78.0	78.0	90.0
ally	C	polynomial	1	0.09	1	66.8 ±4.6	78.4	62.0	62.0
arrive	B	polynomial	1	0.023	1	72.8 ±5.2	74.8	60.0	64.0
breathe	C	polynomial	1	0.027	1	72.3 ±4.3	81.1	80.0	82.0
claim	A	polynomial	2	0.026	1	83.0 ±4.2	94.0	82.0	86.0
cool	C	polynomial	1	0.01	1	63.4 ±4.2	69.3	50.0	56.0
crush	C	polynomial	1	0.03	1	51.4 ±5.8	70.0	46.0	52.0
cry	B	polynomial	1	0.06	1	74.4 ±2.7	83.2	58.0	62.0
deny	B	polynomial	1	0.04	1	61.3 ±4.7	73.0	52.0	58.0
enlarge	C	polynomial	1	0.07	1	80.7 ±3.5	88.3	66.0	76.0
enlist	C	polynomial	1	0.024	1	83.4 ±2.8	91.0	70.2	76.6
forge	C	polynomial	1	0.06	1	52.3 ±5.4	74.3	56.0	60.0
furnish	C	polynomial	1	0.024	1	69.7 ±5.5	80.3	74.0	82.0
hail	C	polynomial	1	0.04	1	83.4 ±4.5	90.3	88.0	90.0
halt	C	polynomial	1	0.08	1	86.8 ±2.4	93.2	86.0	88.0
part	C	polynomial	1	0.06	1	72.0 ±2.1	86.7	60.0	68.0
plough	C	polynomial	1	0.05	1	70.8 ±3.0	89.2	60.0	62.0
plug	C	polynomial	2	0.04	1	59.0 ±4.1	87.7	43.5	47.8
pour	C	polynomial	1	0.07	1	56.0 ±4.7	79.7	62.0	66.0
say	A	polynomial	1	0.04	1	90.6 ±2.3	93.0	88.0	88.0
smash	C	polynomial	1	0.08	1	69.7 ±4.3	86.3	60.0	64.0
smell	C	polynomial	1	0.09	1	62.0 ±4.2	75.0	56.0	62.0
steer	C	polynomial	1	0.04	1	43.7 ±7.2	70.7	50.0	56.0
submit	B	polynomial	2	0.026	1	85.2 ±2.9	90.4	90.0	92.0
swell	C	polynomial	1	0.024	1	50.3 ±6.5	70.3	48.0	50.0
tell	A	polynomial	1	0.04	1	78.4 ±2.0	85.4	82.0	82.0
throw	B	polynomial	1	0.022	1	42.0 ±2.2	63.9	52.0	56.0
trouble	C	polynomial	1	0.04	1	69.0 ±4.0	80.3	76.0	80.0
wake	C	polynomial	1	0.09	1	77.7 ±3.9	88.7	78.0	86.0
yield	B	polynomial	1	0.028	1	52.7 ±5.0	70.7	46.0	46.0
A						87.8 ±2.4	91.8	86.4	86.8
B						64.1 ±4.1	74.3	58.8	62.8
C						68.8 ±4.3	82.1	66.9	71.8
Celkom						83.6 ±2.7	89.0	81.6	82.8

Tabuľka D.28: Výsledky modelu **BestAU** na testovacích údajoch. Modely klasifikátorov boli natrénované pomocou 65 rysov z množiny BestAU, pomocou algoritmu SVM.

Dodatok E

Obsah CD-ROM

V tejto časti popisujeme obsah CD-ROM, ktorý je pripojený k diplomovej práci. V hlavnom adresári disku sa nachádzajú tieto súbory a adresáre:

- `README.txt` Obsahuje tento popis pripojeného CD-ROM.
- `data/` Adresár obsahuje kolekciu VPS. Skladá za z dvoch adresárov:
  - `perl_files/` Obsahuje všetky konkordancie kolekcie VPS vrátane všetkých informácií extrahovaných z dodaných zdrojových súborov. Súbory sú uložené v internom údajovom formáte Perlu a sú k dispozícii pre 32bitové i 64bitové architektúry.
  - `source_files/` Adresár obsahuje kolekciu VPS vo forme zdrojových súborov popísaných v časti 5.4. Skladá sa z nasledujúcich adresárov:
    - `annotated_data/` Obsahuje referenčné vzorky anotovaných konkordancií pre každé sloveso.
    - `confusion_matrices/` Obsahuje matice konfúzie pre multianotované vzorky konkordancií.
    - `morfo/` Obsahuje výstup morfologickej analýzy.
    - `multiannotated_data/` Obsahuje multianotované vzorky konkordancií.
    - `nered/` Obsahuje výstup NER.
    - `parsed/` Obsahuje výstup syntaktickej analýzy.
- `stats/` Adresár obsahuje kolekciu dokumentov vytvorených v tabuľkovom procesore LibreOffice. Obsahujú podrobné štatistiky experimentov, ktoré sme v práci vykonali. Pomenovanie súborov zodpovedá názvom experimentov.
- `text/` Adresár obsahuje text diplomovej práce.
- `tools/` Adresár obsahuje kolekciu skriptov v jazyku Perl a v jazyku R, ktoré sme implementovali v rámci diplomovej práce. Slúžia na prípravu inštancií, extrakciu rysov a evaluáciu výsledkov.