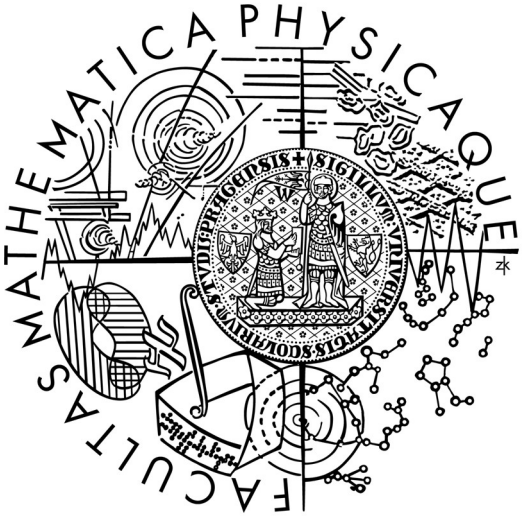
**Univerzita Karlova v Praze**

Matematicko-fyzikální fakulta

**BAKALÁŘSKÁ PRÁCE**




*Josef Čech*

# Oborová klasifikace textu

*ÚFAL - Ústav formální a aplikované lingvistiky*

Vedoucí bakalářské práce: Mgr. Jan Raab

Studijní program: Informatika*, programování,*
*2010*

Na tomto místě bych rád poděkoval svému vedoucímu práce Mgr. Janu Raabovi za ochotu a trpělivost při vedení. Také mu děkuji za poskytnutá učební data, která jsou zdrojem informací při analyzování textu.

Prohlašuji, že jsem svou bakalářskou práci napsal(a) samostatně a výhradně s použitím citovaných pramenů. Souhlasím se zapůjčováním práce.

V Praze dne *27.5.2010* Josef Čech

Název bakalářské práce: *Oborová klasifikace textu*
Autor: *Josef Čech*
Katedra (ústav): Ú*FAL - Ústav formální a aplikované lingvistiky*
Vedoucí bakalářské práce: *Mgr. Jan Raab*
e-mail vedoucího: *raab@ufal.mff.cuni.cz*

Abstrakt: Práce se zabývá porovnáváním textu a jeho kategorizaci. Kategorie, které je program schopen určit, získává v módu učení. Porovnává několik možných algoritmů, které lze využít ke kategorizaci textu. Jde především o Bayesovský model, klasifikaci pomocí neuronových sítí a vektorový model. V praktické části je implementován vektorový model, který využívá kosinovu míru podobnosti. Extrakce termínu vychází z Luhnovy myšlenky o významovosti slov. Jako hlavní zdroj vah pro kosinovu míru podobnosti je využívána hlavně metoda tfxidf s penalizacemi.

Klíčová slova: vektorový model, kosinova míra podobnosti, kategorizace dokumentů

Title: *Branch classification of text*
Author: *Josef Čech*
Department: UFAL - *Institute of Formal and Applied Linguistics*
Supervisor: *Mgr. Jan Raab*
Supervisor's e-mail address: *raab@ufal.mff.cuni.cz*

Abstract: This thesis follows up text categorization. In the first part are described several chosen algorithms for a categorization of documents - the Bayesian model, a categorization with a neural networks and a vector model. Practice part is focused on a algorithm vector model. The vector model is based on idea of two vectors. One vector represents a pattern and second a query. In our case first vector corresponds with a category and the second one with the document. Coordinates of the vector are weights of single words in the text or in the branch depends on, which vector we think about. For comparing are possible to use several procedures like Dice coefficient similarity, Jaccard coefficient or cosine similarity. In my thesis is used cosine similarity. Computing weights is based on frequency of the term in the document and on frequency of documents, which contain the term. Relevant terms are selected on Luhn simple ideas of significance words.

Keywords: Vector model, cosine similarity, Bayesian model, neural network

# Obsah

# Úvod

Kategorizace textu je poměrně nová disciplína, která má však jistě své místo v počítačové lingvistice, a to hlavně díky rozvoji informačních technologií. Stále více informací je ukládáno do digitální podoby, ať už jde o digitalizaci papírových materiálů nebo články a dokumenty, které vznikly už v digitální podobě.

V současné době neexistuje systém, který by byl schopen se 100% úspěšností analyzovat text jakýmkoliv způsobem. Hlavní příčinou tohoto neúspěchu je nejspíše předmět zkoumání, tím je přirozený jazyk.

Přirozený jazyk je těžko popsatelný formálními metodami. Úkol, tedy porozumění přirozenému jazyku, by pro 100% úspěšné řešení vyžadoval analýzu textu na rovině, která se rovná lidskému porozumění textu. Tedy nejen porozumění významům slov, ale i vět a vztahů mezi slovy a větami. Přitom neexistuje algoritmus, který by zajišťoval bezchybné porozumění jednotlivým slovům
v textu. To znamená porozumění na všech jazykových rovinách (viz 2.12). Při zpracování dochází k problémům již na úrovni předzpracování, kdy ze znaků vytvoříme slova s interpunkcí. Toto rozdělení na slova nelze popsat jednoduchou sadou pravidel. Nejednoznačnost je způsobena hlavně definicí slova (2.1), která neříká jak přesně chápat slovo.

Díky výše zmíněným důvodům je současná počítačová lingvistika více orientovaná na statistické vlastnosti jazyka. Vznikají jazykové korpusy (rozsáhlé soubory textů s různými metajazykovými značkami, které podávají informace o typu textů a kategorizaci slov) a řízené slovníky, z kterých jsou čerpány statistické informace. Výhodou statistického přístupu je, že v případě změn v jazyce (v krátkém intervalu 1961[1] – současnost nebo 1994[2] -současnost lze předpokládat hlavně změny použití slov), se tyto změny mohou promítnout do statistických dat.

Cílem práce je vytvořit program, který bude schopen analyzovat textový dokument. Na základě této analýzy bude odhadnut obor. Obor bude určen pomocí statistik výskytu z trénovacích dat. Přesné obory, které bude program schopen rozlišovat, jsou dány trénovacími daty. Kategorie bude získávána z trénovacích sad dokumentů.

K tomuto účelu byl implementován vektorový model s váhovou funkcí *tfxidf*. Vektorový

1 Vznik první (Brownova) korpusu
2 Založení Českého národního korpusu pod vedením prof. Františka Čermáka

model porovnává dva aspekty dokumentů - jednotlivá slova a sousloví. Hlavní kategorizační kritérium je váha slov, váhy sousloví pouze zpřesňují informaci. Porovnávacím kritériem mezi kategorií a dokumentem je kosinova míra.

## 1.1 Současný stav automatické kategorizace

V současné době je potřeba třídit informace skutečně nutná. Vektorový model společně s váhovou funkcí *TDxIDF* a kosinovou mírou je jeden z jednodušších typů kategorizace. Nyní se do popředí dostává model, který je založený na podobném principu, SVM - Support Vector Machines. Tento model dosahuje výborných výsledků, a to především při binárním rozhodování, tzn. zda dokument patří do kategorie. Tento model vychází z představy, že jednotlivé příklady se rozdělí na dvě kolekce bodů, např. pozitivní a negativní. Cílem algoritmu je poté najít ideální oddělovač mezi těmito dvěma kolekcemi. Nejbližší datové body vzhledem k oddělovači pak slouží ke kategorizaci dokumentu. Výhodou této metody je především rychlost. Pro správnou kategorizaci jí stačí podstatně méně bodů (dokumentů), než obsahují trénované kolekce. Nevýhodou ovšem je, že je optimalizovaná pro dvě kategorie. I tak se úspěšnost pohybuje kolem 90 %. Podobně úspěšné výsledky má i metoda k-NN (metoda *k* nejbližších sousedů). Tento klasifikační algoritmus je založen na myšlence, že klasifikovaný objekt převezme třídu dle třídy objektů, kterých je nejvíce a jsou nejblíže (myšleno v nějaké metrice, např. Euklidova nebo Hammingova).

Ve spojení metody k-NN, váhové funkce TFxIDF a kosinovou mírou bylo dosaženo v průměru úspěšnosti 92.32 % (při použití Jaccardovy a Diceho míry a při použití WIDF byla úspěšnost řádově o procenta nižší) (Wa`el Musa Hadi a kol., 2007). Pro testování algoritmu bylo sebráno 20 000 dokumentů, které byly rozděleny do 20 kategorií. Testováno bylo několik verzí algoritmu, které se lišily ve váhové funkci (*TFxIDF*, *WIDF*[3]) a výběru funkce míry (Jaccardova, Diceho a kosinova míra).

## 1.2 Struktura práce

Práce je rozdělena do tří částí. První část tvoří seznámení se s danou problematikou. Objasňuje problémy, které spočívají v analýze textu a extrakci dat z jednotlivých dokumentů.

3 WIDF(d,t) - Weighted Inverse Document Frequency - Vážená inverzí dokumentová frekvece. Jedná se o poměr frekvence termínu t v dokumentu d ku frekvenci termínu t v sadě dokumentů D

$$WIDF(d,t) = TF(d,t) / \sum_{(i \in D)} TF(i,t)$$

Popisuje, jaké jsou možnosti předzpracování dat, a nabízí i několik algoritmů, které je možno alternativně na tento úkol použít. Jedná se především o kapitoly 2, 3, 4.

Druhá část práce je seznámením s konkrétním řešením programu. Jedná se o postupy, kterými byly řešeny úkoly, které vznikaly pří vytváření programu. V této části se také případný uživatel programu seznámí s jeho ovládáním a možnostmi. Tato část je reprezentována kapitolou „Implementace“ a kapitolou „Uživatelská příručka“.

Závěrečná část se týká hlavně zhodnocení výsledků a celkového zhodnocení tvorby programu. Jsou zde navrženy další možná vylepšení programu nebo změny řešení některých dílčích problémů. S touto částí jsou hlavně spojeny kapitoly „Interpretace výsledků“ a „ Závěr“.

# 2 Slovníček pojmů

## 2.1 Slovo

Slovo bývá považováno za základní jednotku lexikální roviny. Přestože jde jednu ze základních jednotek, není v jazykovědě dosud vytvořena uspokojivá jednoznačná definice. Stručně lze slovo definovat jako ustálenou jednotku jazyka, která je tvořena fonémem nebo řadou fonémů a je ve větě přemístitelná. Nese lexikální nebo (a) gramatický (i pragmatický) význam.

V širším smyslu lze slovo popsat pomocí následujících základních charakteristik[4] :

1) Formálně-významová jednota
2) Fonetická a fonologická utvářenost
3) Grafická podoba
4) Významová stránka
5) Přemostitelnost a nahraditelnost
6) Ustálenost

## 2.2 Levenshteinova metrika

Pomocí Levenshteinovy metriky lze porovnávat dva řetězce. Je definována jako minimální počet operací mazání, vložení a nahrazení.

Příklad: V Levenshteinově metrice je mezi slovy švec a ševce vzdálenost 3:

a) švec – švvc – operace nahrazení e => v
b) švvc – ševc – operace nahrazení v => e
c) ševc – ševce – operace vložení e

## 2.3 Lemma

Lemma je základní (slovníkový) tvar. V různých jazycích existují různé konvence pro označení tvaru jako lemmata.

V češtině je to pro substantiva, adjektiva, pronomina a numerale tvar nominativu singuláru.

4 Vice informací v [2]

Pro verba je tímto tvarem infinitiv. Adverbia jsou reprezentována tvarem pozitivu. Zbylé slovní druhy jsou neohebné, mají proto pouze jeden tvar.

## 2.4 Automatická indexace

Indexace je proces vyjádření výsledku analýzy dokumentu prostřednictvím prvků selekčního jazyka nebo přirozeného jazyka, obvykle s cílem umožnit zpětné vyhledávání.

Automatická indexace je vyjádření obsahu dokumentu pomocí automatického výběru slov nebo termínů z textu nebo pomocí automatického přiřazování termínů selekčního jazyka. Běžně se automatická indexace dělí do dvou základních skupin:

1) Automatická extrakce *(slovní indexace*)

Indexovací metoda je založena na výběru termínu přímo z plného textu dokumentu. Využívá pouze přirozeného jazyka.

2) Automatické přiřazování – (*pojmová indexace*)

Indexovací metoda využívající řízené slovníky nebo také znalostní báze, která porovnává výrazy
z plného textu dokumentu

V současné době nejsou i ty nejsofistikovanější metody stoprocentně přesné. Daleko lepších výsledků je dosahováno v systémech, ve kterých je automatická indexace podporována člověkem (intelektuální indexace, human indexing). Algoritmy pro automatickou indexaci jsou připraveny potencionální indexované termíny, které jsou poté kontrolovány člověkem.

## 2.5 Zipfův zákon

### 2.5.1 Zipfův zákon

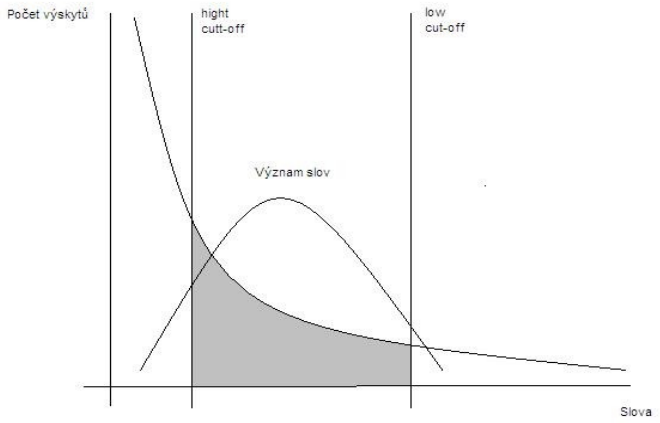
George Kingsley Zipf (1902-1950) přišel s myšlenkou, že pokud budeme uvažovat f jako frekvenci slova a r jako jeho pořadí, produkt f.r pro každé slovo je přibližně konstanta. Tuto myšlenku potvrzuje např. Brownův korpus, kde tvoří slovo s nejvyšší frekvencí 7% korpusu, slovo s druhou nejvyšší frekvencí 3,5%.

### 2.5.2 Luhnův předpoklad

H. P. Luhn (1886-1964) předpokládá, že největší schopnost charakterizovat dokument, a

tím i největší význam, mají slova se střední frekvencí. Pokud bychom si představili graf výskytu frekvencí, slova, která mají největší frekvenci, jsou brána jako "nutný šum" a slova s nejmenšími frekvencemi jako slova nevýznamová. Třetí skupinou jsou tzv. významová (signifikantní) slova. Ta jsou vymezena dvěma zářezy (cut-offs). Hodnoty těchto zářezů nejsou jednoznačně určeny a musí se pro každý obor určit pomocí statistických metod.

Obr. 1 – Grafické znázornění Luhnova předpokladu




Luhn obhajuje tuto tezi na základě způsobu, jakým je text vytvářen. Pokud totiž mluvíme o nějakém objektu, zřídka kdy se stává, že se o tomto objektu zmíníme pouze jednou. I když je snaha používat synonyma, je zásoba synonym omezena. Nakonec se i synonyma začnou opakovat.

## 2.6 Podmíněná pravděpodobnost

Jde o zpřesnění apriorní pravděpodobnosti náhodného jevu A za předpokladu, že nastal jev B, většinou je značena jako P(A|B). Matematicky je definována $P(A|B) = P(A \cap B) / P(B)$ jako Čitatel je interpretován jako pravděpodobnost události současného výskytu událostí A a B. Jmenovatel je pravděpodobnost výskytu jevu B za podmínky $P(B) > 0$ .

## 2.7 Morfém

Morfém je nejmenší vydělitelná součást slova, která má svůj lexikální, gramatický nebo lexikálně-gramatický význam.

## 2.8 Bayesův teorém (Bayesova věta)

Toto pravidlo je používáno v teorii pravděpodobnosti. Udává, jak podmíněná pravděpodobnost jevu A při výskytu B souvisí s opačnou podmíněnou pravděpodobnosti. Pokud A a B jsou náhodné jevy a P(A) a P(B) a P(B) > 0, lze tuto větu vyjádřit jako $P(A|B)=P(B|A)P(A)/P(B)$ .

## 2.9 B-strom

B-strom je datová struktura určená k vyhledávání. Obecně si pod pojmem strom přestavíme graf, který neobsahuje kružnici. Nejednodušší je tzv. binární strom, který v každém svém uzlu obsahuje jednu hodnotu nebo podmínku, která reprezentuje vyhledávací klíč. Z takovéhoto uzlu vedou dvě hrany do dalších dvou uzlů.

B-strom se liší v počtu vyhledávacích klíčů, které jsou určitým způsobem uspořádány (pro textové hodnoty je to např. lexikografické uspořádání). Počet takovýchto vyhledávacích klíčů v jednom uzlu je libovolný. Pomocí těchto klíčů se vyhledávají konkrétní záznamy.

## 2.10 Morfologická analýza a tag

Morfologická analýza je proces, jehož výsledkem je přiřazení morfologických kategorií a jejich hodnot ke tvaru slova. Např. číslo je morfologická kategorie zatímco singulár (jednotné číslo) je jeho hodnota.

Při počítačovém zpracování je zavedena množina značek (tagset), které tyto kategorie reprezentují. Pro zápis takových množin je možno uvažovat několik notací. Nejčastěji používaná (i v tomto programu) je notace poziční. V této notaci je hodnota kategorie (reprezentovaná značkou) vložena na místo, které odpovídá této kategorii. Dále oproti obecné verzi morfologické analýzy je při počítačové morfologické analýze důležitá tzv. lemmatizace.

Tag je uspořádaný soubor značek pro jeden slovní tvar.

## 2.11 Jazykové roviny

V teoriích zpracování jazyka se jeho popis dá rozdělit do několika základních rovin:

1) rovina fonetiky / pravopisu

2) rovina fonologie, morfonologie
3) morfologická rovina
4) rovina syntaxe (povrchová)
5) sémantická rovina (hloubková syntax)
6) pragmatická rovina

Pro praxi lze některé roviny sloučit (skládání morfonému a pravopisná pravidla lze realizovat najednou) nebo úplně vynechat (např. většina základních vstupů je v textové podobně, a proto není potřeba analyzovat fonémy).

V praxi se používá zjednodušený systém rovin:

1) předzpracování
2) morfologická
3) analytická (syntaktická)
4) tektogramatická rovina

### 2.12 Ostatní termíny

Pro lepší porozumění níže popisovaných algoritmů jsou vhodné i některé základnější pojmy z matematiky. Protože jde o téma na hranici mezi jazykem a matematikou, může být tato práce zajímavá i pro lingvisty, pro které tyto pojmy mohou být neznámé. Proto zde uvádím seznam těchto pojmů a zároveň připisuji i možné zdroje, kde je možné tyto pojmy nastudovat.

Jedná se o pojmy: *Vektorový prostor* (Rohn, 2004) , *Graf , Orientovaný graf* (Ryjáček et al., 2004)

# 3 Analýza problému

## 3.1 Předzpracování vstupních dat

Zpracování vstupních dat je první fází. V této části procesu je důležité získání potřebných informací z předložených dat (obvykle dokumentu). Při úkolu kategorizace dokumentů, ale i při jiných úkolech týkajících se analýzy textu (např. indexace, sumarizace atd.), je sada úkolů představující analýzu vstupních dat myšlenkově velmi podobná, ale konkrétní řešení analýzy vstupních dat se liší podle hlavního využití celého programu. Např. při indexaci dokumentů záleží na frekvencích, resp. vahách, které jsou jednotlivým slovům přidělovány.

Pro konkrétní řešení, tedy kategorizaci dokumentů, je úkolem analýzy vstupních dat zpracovat data do tvaru, kterému odpovídají vstupy použitých **kategorizačních metod.** Obecně lze říci, že pro zpracování textu existuje v první fázi celkem jednotný postup. Tento postup je často spojován hlavně s typem automatické indexace – automatickou extrakcí. Tento proces lze rozdělit na 3 části:

1) lexikální analýza
2) lemmatizace
3) vážení

### 3.1.1 Lexikální analýza

Lexikální analýza je první a nejjednodušší proces při práci s textem. Tento proces má za úkol vytvořit ze vstupního souboru posloupnost slov, která budou dále procházet analýzou. Celý proces je možné rozdělit do tří úkolů. Prvním je vytvoření seznamu slov, druhým vyloučení nevýznamových a nespecifických slov a třetím nalezení sousloví.

Relativní jednoduchost je zajištěna definicí slova v přirozeném jazyce[5]. Nejjednodušší definice říká, že slovo je řetězec znaků mezi dvěma mezerami. Tímto lze proces lexikální analýzy ve zjednodušené podobě praktikovat pouze sledováním mezer mezi slovy. Slovo jako lingvistickou jednotku lze jen těžko přesně nadefinovat (viz 2.1). Problematické jsou zkratky, výrazy spojené spojovníkem a číslovky.

Zkratky jsou obtížné, protože obsahují tečku, kterou je nutno odlišit od tečky větné. U

5 Vztahuje se pouze pro slova v jazycích používajících psaných latinkou nebo azbukou (viz 2.1)

výrazů se spojovníkem je nutno rozlišit, zda takto spojený výraz považovat za jedno nebo za dvě slova.

U číslovek je nutné rozhodnout, zda budou zpracovávány jako slova, nebo budou vypouštěny.

Vyloučení nevýznamových a nespecifických slov se často děje pomocí negativního slovníku. Negativní slovník tvoří především slova gramatická (nemají vlastní sémantický význam), tedy slova patřící k určitým slovním druhům (např. předložky a spojky). Na základě empiricky podloženého předpokladu (viz 2.5.1), že nespecifická a nevýznamová slova i sousloví mají vyšší frekvenci výskytu než slova významová, patří do negativního slovníku také slova s vysokou absolutní (i relativní) frekvencí (takové výrazy chápeme jako příliš obecné). Často využívaným předpokladem je, že nevýznamová a nespecifická slova jsou krátká. Pokud bychom takto slepě vyřazovali krátká slova, existuje možnost, že vyřadíme důležitý termín. Pokud je použit tento předpoklad, je zároveň vytvářen také anti-negativní slovník, který obsahuje právě krátké termíny.

Obtížnější částí úkolu lexikální analýzy je schopnost identifikovat sousloví. Dvojslovný indexovaný termín má vyšší selektivní význam než jednoslovný termín (viz 2.3). Pro identifikací sousloví existuje několik technik, uvádím dvě nejobvyklejší:

**a) Statistická identifikace sousloví** – u této metody se vychází z frekvence daného **sousloví** (záleží na pořadí slov) a na vzdálenosti slov v textu. Tato vzdálenost může být určena počtem slov mezi slovy daného sousloví nebo můžeme sledovat současný výskyt dvou slov ve větě, odstavci nebo jiné ucelené části textu. Častý výskyt dvou (a více) slov blízko sebe, ještě nemusí znamenat, že takováto slova tvoří sousloví, proto není tato metoda stoprocentně úspěšná.

**b) Syntaktická identifikace sousloví** - identifikace probíhá jako v předchozím případě s tím rozdílem, že v potencionálním sousloví je analyzována také jejich syntaktická složka.

### 3.1.2 Lemmatizace

Lemmatizaci neboli přiřazení základního tvaru lze provádět na základě dvou odlišných přístupů.

První vychází z pravidel tvarosloví jazyka a seznamu kořenů (radix), předpon (prefix), přípon (infix) a koncovek (suffix). Na základě těchto dat lze teoreticky dogenerovat (derivovat) každý tvar slova. Opačným způsobem lze odhadnout, jak bylo slovo utvářeno. Slabinou tohoto přístupu jsou výjimky (nom. Zeus – gen. Dia).

Druhým způsobem je identifikace na základě slovníku. Slabinou tohoto přístupu je hlavně mnohotvárnost a vývoj jazyka. Tyto faktory přináší nové významy starých tvarů (např. před rokem 1963 mělo slovo myš jediný význam, a to hlodavec) nebo úplně nová slova a slova, která se přelévají z **obecné češtiny** do **spisovné češtiny** (např. outsourcuji). Tyto nedostatky se poté musejí řešit stálou aktualizací slovníku pomocí nových dat, které byly ošetřeny a analyzovány člověkem. Tímto způsobem jsou ale pokryta pouze slova ve slovníku. Jazyk se neustále vyvíjí, proto je možná množina slov pro slovník teoreticky nekonečná a sběr dat představuje dlouhý a téměř nekonečný proces.

Nejúspěšnější a pro češtinu nejvyužívanější metodou je lemmatizace spojená s statistickým taggingem. Tagging je v tomto případě založen na relativních četnostech značek. Tento způsob má chybovost pro češtinu zhruba 5% (Hajič, 2001).

Na první pohled by se mohlo zdát, že transformace slovního tvaru na slovníkový tvar má význam pouze v případě algoritmů a metod spolupracujících se slovníkem (abychom je mohli rozškatulkovat). Tato představa je mylná, alespoň v případě kategorizace dokumentů.

Dle Bachmanové (BACHMANOVÁ et al, 2002) je kořen nejstabilnějším prvkem slova a obsahuje jádro významu. Plný význam je poté dotvořen pomocí afixů (předpon a přípon) a vzniká kmen slova. Lemma jako vybraný tvar slova obsahuje celý kmen slova. Díky tomu neztratíme sémantickou informaci, kterou transformované slovo neslo. Rozdělení slov pod jednotlivá lemmata je pouze druhotným jevem.

Na konci lemmatizace máme zjednodušeně řečeno seznam lemmat s počtem jejich výskytů v dokumentu.

### 3.1.3 Vážení

Vážení (weighting) je zobrazení *w: T -> S, kde T je získaná množina lemmat a S je podmnožina R.* Často S = <0,1).

Tento proces přiřazuje každému termínu ze seznamu získaného lemmatizací určitou hodnotu, která vyjadřuje důležitost a schopnost termínu charakterizovat dokument.

Vážení termínů je funkce, která je ovlivněna několika faktory. Mezi základní patří:

1) vnitřní faktory – bereme v úvahu samotnou povahu slova, např. roli hraje slovní druh
2) faktory stojící mimo samotný termín – tím může být např. délka textu, počet různých termínů v textu
3) faktory popisující vztah mezi termínem a textem - do této kategorie patří především

frekvence termínu. Dalšími faktory tohoto typu mohou být: pozice termínu v textu, kontext termínu

4) globální faktory – jedná se o vztah mezi termínem a databází. Typickým příkladem je frekvence termínu v databázi.

Na základě těchto faktorů můžeme intuitivně vytvořit některá dílčí kritéria, která budeme dále využívat ve váhových funkcích. Jsou to hlavně:

1) f(t,T) frekvence termínu t v textu T (článku)
2) n(t) počet dokumentů, ve kterých se objevuje termín t
3) F(t) počet všech termínů t v databázi
4) f(T) počet všech termínů v textu T
5) k(T) počet všech různých termínů v textu T
6) N počet dokumentů v databázi

Nyní můžeme přistoupit k samotnému výčtu váhových funkcí, které budou využívat právě výše zmíněné parametry. Tento seznam samozřejmě není konečný a váhové funkce je možné kombinovat.

Přehled základních váhových funkcí:

1) Prostá frekvence - tato váhová funkce zohledňuje absolutní výskyt termínu t v textu T; w(t,T) = f(t,T)
2) Logaritmická frekvence – zohledňuje fakt, že x-násobný výskyt neznamená x-násobnou důležitost. V úvahu se berou pouze řády výskytů; w(t,T) = log (f(t,T))
3) Normalizovaná frekvence – eliminuje délku textu; w(t,T) = f(t,T)/f(T)
4) Relativní frekvence – zohledňuje celkový výskyt termínu v databázi; w(t,T) = f(t,T)/F(t)
5) Selektivní síla – tato funkce není přímo váhovou funkcí, slouží ke stanovení míry, s jakou je indexovaný termín schopen rozlišit dokument. Nízká hodnota značí malou selektivní sílu, vysoká hodnota naopak vysokou selektivní sílu. w(t) = N – n(t) / N

### 3.1.4 Učení

Konkrétní učební metody závisí na vybraném konkrétním algoritmu. V případě Bayesova klasifikátoru (viz 3.2.1) jsou to pravděpodobnosti výskytů slov v dokumentech, v případě

neuronových sítí (viz 3.2.2) je to nastavování vah mezi jednotlivými neurony.

Obecně lze říci, že strojové učení pro tento typ úkolu bude vycházet z termínů a jejich vah, tedy z frekvencí těchto termínů v textu. Instinktivně se nabízí obyčejný nebo vážený průměr vah těchto termínů.

## 3.2 Kategorizace dokumentů

Jako všechny práce s dokumenty a knihami je lidská kategorizace velmi náročná myšlenkově i časově. Ve většině dnes používaných kategorizačních metod jsou nedůležitější tzv. charakteristické (indexované) termíny. Termíny jsou pro jednotlivý dokument nejčastěji vybírány na základě jejich frekvence v dokumentu. Při výběru termínů ze sady dokumentů charakterizujících kategorii je nutné přihlédnout i k tomu, v kolika dokumentech se slovo vyskytovalo.

Hlavním rozdílem mezi systémy implementujícími kategorizaci dokumentů jsou použité algoritmy (více 2.2) a sběr dat (viz 2.1.4 a 3.6.).

### 3.2.1 Srovnání kategorizačních algoritmů

3.2.1.1 Bayesův klasifikátor (Bayesův model)

Tento model je založen na Bayesově teorému a na předpokladu, že výskyt jednoho slova nepodmiňuje výskyt jiného. Např. pokud se vyskytne slovo „ovoce“, ze zkušenosti víme, že můžeme očekávat v jeho okolí slova jako „jablko“ nebo „zraje“. Naopak neočekáváme termíny jako „počítač“ nebo „televize“. Z pravděpodobnostního hlediska lze tyto zkušenosti vyjádřit zvýšenou nebo sníženou pravděpodobností výskytu nalezeného a předpokládaného termínu. V praktických testech se tento postup moc neužívá, protože nemá příliš velký vliv. Proto je tento model také někdy nazývaný „naivní Bayesův model“.

Fází učení tohoto modelu je získání pravděpodobností výskytů jednotlivých termínů v kategoriích $C$ ($C$ je množina všech kategorií, kterými můžeme klasifikovat dokument D) a také používání kategorií $C_i$ v celé zkoumané databázi.

Model vychází z podmíněné pravděpodobnosti. Na základě výskytu jevu $X$ nastane jev $C_i$. Pokud je proměnná $X= \{x_1, ...,x_n\}$ chápána jako množina termínů v analyzovaném dokumentu a proměnnou $C_i$ lze chápat jako objekt (kategorii), lze vzorec přepsat jako:

$P(C_i|X)=P(X|C_i)/P(X)$ . *P(Ci | X)* je pravděpodobnost, že jde o kategorii *Ci* za

předpokladu, že byly nalezeny termíny *X*. *P(X | Ci)* je naopak pravděpodobnost, že nalezneme termíny *X* v dokumentu kategorie $C_i$.

$$P(X|C_i)/P(X)=P(x_{1,}x_2\dots,x_n|C_i)/P(x_{1,}x_2\dots x_n)$$

Nyní budeme zanedbávat jmenovatele, protože se jedná u každého analyzovaného dokumentu o konstantu, která se dá vyjádřit jako $P(x_{1,}x_{2,}..x_n)=\prod P(x_i)$ , protože výskyt jednotlivých termínů je považován $P(x_{1,}x_2\dots,x_n|C_i)*P(C_i)=P(C_i,x_{1,}x_2\dots,x_n)$ za nezávislý

Pravděpodobnost, s jakou termíny $x_1,x_2,..,x_n$ patří k objektu (kategorii) $C_{i,}$ lze vyjádřit jako:

$$P(C_i,x_{1,}x_2\dots,x_n=P(x_{1,}x_2\dots,x_n|C_i)*P(C_i))$$
$$P(C_i,x_{1,}x_2\dots,x_n)=P(x_2\dots,x_n|C_i)*P(C_i)*P(x_1|C_i)$$
$$\dots$$
$$(C_i,x_{1,}x_2\dots,x_n)=P(C_i)\prod_{(j\le n)}^{n}P(x_j|C_i)$$

Pokud budeme za množinu tříd považovat dvě nezávislé třídy např. při filtraci spamu – *S* (spam), ¬*S* (non-spam) a sadu slov *T* vybraných z dokumentu *D,* bude poté problém klasifikace redukován na pravděpodobnosti

$$|T|=n-počet\ slov$$
$$P(S|T)=P(S)\prod_{(j\le n)}^{n}P(t_j|S)$$
$$P(\neg S|T)=P(\neg S)\prod_{(j\le n)}^{n}P(t_j|\neg S)$$

Po výpočtu těchto pravděpodobností následuje vlastní rozhodnutí, které lze získat porovnáním *P(S|T) > p(¬S|T).* V tomto konkrétním případě by bylo rozhodnuto, že dokument *D*, ze kterého byly termíny *T* extrahovány, patří ke kategorii S (tedy spam). Lze to různě optimalizovat, např. převést na logaritmus:

$$P(S|T)>P(\neg S|T)$$
$$\ln(P(S|T)/p(\neg S|T))>0$$
$$\ln(P(S)*\prod_{j}^{n}P(t_j|S)/P(\neg S)*\prod_{j}^{n}P(t_j|\neg S))>0$$
$$\ln(P(S)/p(\neg S))+\ln(\prod_{j}^{n}P(t_j|S)/\prod_{j}^{n}P(t_j|\neg S))>0$$

Pro rozšíření na multikategoriální klasifikátor se logicky nabízí kategorii ¬*S* chápat jako zbylé kategorie $s_1,..,s_n$:

$$P(\neg S) = P(s_1 \cup s_2 \cup ... \cup s_{(n-1)}) = \sum_{i}^{(n-1)} P(s_i) = 1 - P(s_n),\ kde\ s_n = S$$

$$P(\neg S \mid T) = \sum_{i}^{(n-1)} P(s_i \mid T) = 1 - P(S \mid T)$$

Pokud můžeme předpokládat, že kategorie nejsou na sobě nijak závislé, je výše uvedený výraz pravděpodobnost, s jakou patří do zbylých kategorií.

Existuje také možnost, že nebudeme muset procházet všechny kategorie. Mějme kategorii *K* a kategorie *C= {c_i}* pro všechna *i* a $c_i \neq K$, *D* je analyzovaný dokument. Z pozorování lze usoudit, že *ln(p(D|K)/p(D|C)) > 0 => D* patří do kategorie K.

Idea důkazu tohoto výroku je velmi jednoduchá. Vycházejme z předpokladu, že kategorie jsou nezávislé. Podle teorie pravděpodobnosti je součet pravděpodobností v jevovém prostoru roven 1. Při aplikaci na aktuální problém je jevem příslušnost množiny X z dokumentu D do kategorie $C_i$. V implikaci předpokládáme, že platí *ln(p(D|K)/p(D|C)) > 0 => D patří do kategorie K (značení vychází z předešlého odstavce)*, ale nejvyšší pravděpodobnost získala kategorie L. Lze veškeré poznatky zapsat jako:

$$P_D(K) > P_D(L) + P_D(C_{KL}) - \textit{předpoklad implikace}$$
$$P_D(L) > P_D(K) - \textit{podmínka vyběru jiné kategorie}$$
$$P_D(L) > P_D(L) + P_D(C_{KL}) - \textit{spojení podmínek}$$
$$P_D(K)\ \textit{je pravěpodobnost s jakou dokument D patří ke kategorii K}$$
$$P_D(L)\ \textit{je pravěpodobnost s jakou dokument D patří ke kategorii K}$$
$$P_D(C_{KL})\ \textit{je pravěpodobnost s jakou dokument D patří ke jiné kategorii než K nebo L}$$

V poslední nerovnici se dostáváme do sporu, protože funkce P může mí pouze kladné hodnoty[6].

Slabinou obecného Bayesovského klasifikátoru je rychle rostoucí prostorová náročnost a také zajištění apriorní pravděpodobnosti jednotlivým kategoriím, tedy pravděpodobnosti $p(C_i)$, které závisí na vstupních učebních datech.

Je nutné zajistit vzorek, ve kterém jsou kategorie zastoupeny podobně jako v reálném světě. Pokud bychom dávali nějaké kategorii přednost, může se stát, že díky vysoké apriorní pravděpodobnosti bude klasifikátor nepoužitelný a jeho výsledky neprůkazné.

#### 3.2.1.2 Klasifikace pomocí neuronových sítí

Neuronové sítě jsou struktury známé z umělé inteligence. Jedná se o paralelu s živými

6 pokud bychom uvažovali pouze dvě kategorie, lze dokazovaný výrok prohlásit za ekvivalenci

tvory, jejichž mozek je tvořen neurony. Z matematického hlediska lze neuronové sítě popsat jako **orientované grafy**, které mají několik vstupů a obvykle jeden výstup. Uzel takového grafu se nazývá neuron. Hrany mezi neurony umělé neuronové sítě mají svou váhu (i zápornou). Samotný neuron je specializovaný na zpracování vstupních vah a přenos zpracovávané informace. Z pohledu topologie sítě lze neurony rozdělit do tří typů:

1) vstupní vrstva (neurony) - nejsou mezi nimi žádné hrany a jejich hrany míří jen do dalších skrytých vrstev

2) skryté vrstvy – obsahují o mnoho více neuronů než ostatní vrstvy

3) výstupní vrstva – neurony v této vrstvě mají pouze výstupy. Z této vrstvy se odečítají výsledky.

U umělých neuronových sítí mluvíme o dvou typech učení. První typ je tzv. učení s učitelem. Tento typ simuluje učení živých organismů. Na prázdné umělé neuronové síti (hrany mají automaticky vygenerovanou hodnotu) převedeme trénovací data na vstupy, které přijímá tato neuronová síť. Každý prvek trénovacích dat je převeden na vstupy a je vložen do neuronové sítě. Po průchodu neuronovou sítí porovnáme výsledky s výsledky učitele, tedy zdroje, u kterého nepochybujeme o správnosti výsledku. Nejčastěji jde o lidského analyzátora. Porovnání obou výsledků je pak vyjádřeno v rozdílu, který je použit pro zpětné učení. Tímto rozdílem zpětně ohodnotíme váhy ve skrytých vrstvách neuronové sítě metodou zpětné propagace chyby (Backpropagation).

Druhá metoda učení je analogicky nazývána tzv. učením bez učitele (samoorganizace). Při tomto druhu učení si vstupy utřídí sama neuronová síť, např. je rozdělí do skupin podle podobnosti.

Pokud by k řešení byly použity neuronové sítě, nabízí se možnost představit si vstupy neuronové sítě jako váhy termínů vstupního dokumentu. Výstupy by mohly tvořit váhy, s jakými jsou všechny termíny spojeny s oborem, který je reprezentován výstupním neuronem.

Neuronové sítě se s oblibou aplikují na problémy, které jsou z různých důvodů těžko rozhodnutelné. Jedná se např. o neúplná data (testy s automatickou diagnózou), poškozená data (zpracování obrazu), nemožnost formálního popisu nebo jeho vysoká složitost (srovnávání se vzorem, předvídání).

Neuronové sítě se zdají jako univerzální řešení, ale nejsou vhodné úplně pro všechny úlohy, protože spoje mezi neurony vznikají na základě chyby. Tato chyba se pak rozptýlí do celé

neuronové sítě. Poté záleží, jak jsou neurony pospojovány. Tento způsob trénování může způsobit, že některé výsledky nebudou zcela věrohodné. Tato nedůvěra není na místě pro některá odvětví např. bankovnictví. Tyto poznámky platí pro všechny zde zmiňované klasifikační algoritmy.

Z hlediska implementace je nevýhodou zejména obtížné hledání správných propojení mezi neurony. Učení na neuronových sítích bývá pomalé a existuje možnost, že při vkládání stále stejných dat vzniknou vazby i tam, kde by neměly.

#### 3.2.1.3 Vektorový model

Tento model je klasickou metodou porovnávání a vyhledávání dokumentů na základě dotazu. Základní idea tohoto modelu spočívá v transformaci dokumentů a dotazů do vektorů (tzv. charakteristický vektor), které lze poté porovnávat. Souřadnice takového transformovaného vektoru odpovídají vahám jednotlivých termínů. Vybranou porovnávací metodou poté porovnáváme jednotlivé dokumenty. S vektorovým modelem je často spojována metoda *tfxidf*. Jedná se o váhovou funkci, ve které fungují jako hlavní parametry hodnoty obyčejná frekvence termínu ($tf_i$) a inverzní dokumentová frekvence (*idf*). Tato hodnota je definována jako $idf_i = \ln(1|df_i)$ , kde $df_i$ je počet dokumentů, ve kterých se termín *i* vyskytuje. Definici váhy pro termín *i* lze přepsat jako $w_i = tf_i * \ln(1|df_i)$ . Takováto definice je samozřejmě pouhý základ, který lze upravovat na základě pozorování, jazykových pravidel atd. (více 3.1.3).

Další metoda, která je velmi často spojována s vektorovým modelem, je tzv. koosinova podobnostní metoda (cosine similarity). Pro názorné vysvětlení uvažujme dva charaktDiceůvékoeficient$V_1$ a $V_2$, které mají svůj počáteční bod v počátku souřadnic. Zmíněné vektory svírají úhel α, který může být považován za metriku. Čím menší úhel mezi sebou vektory svírají, tím jsou jejich charakteristiky podobnější. Podle vlastností metriky platí $m(a,b)=0 \Leftrightarrow a \sim b$ . Funkce *cos* α je lehce dopočitatelná pomocí vektorů $V_1$ a $V_2$.

Složením předchozích znalostí lze určit mezní hodnoty:

$$\mathrm{sim}(V_{1,}V_2)=\frac{(\sum_i^n v_{1i}*v_{2i})}{(\sqrt{\sum_i^n v_{1i}^2}*\sqrt{\sum_i^n v_{2i}^2})}=\cos(\alpha)$$

*α je úhel , který svírajíc vektory* $V_1$ *a* $V_2$

$\mathrm{sim}(V_{1,}V_2)=1\Leftrightarrow V_1=V_2$
$\mathrm{sim}(V_{1,}V_2)=0\Leftrightarrow V_1$ *a* $V_2$ *nemají nic společného*

Tato měřící technika patří mezi nejoblíbenější, ale u vektorového modelu lze využít i jiné porovnávací techniky:

1. vnitřní produkt – jedná se pouze o jednoduchý skalární součin charakteristických vektorů $\mathrm{sim}(V_1,V_2)=V_1\dot{V}_2$

2. Jaccardův koeficient (Jaccardova míra) – tato míra je často používána pro porovnání dvou množin. V teorii množin je definován jako $J(A,B)=\frac{(A\cap B)}{(A\cup B)}$ ,kde A a B jsou množiny. Původní použití předpokládá pouze hodnoty 0 a 1, které odpovídají přítomnosti a nepřítomnosti prvku v množině. Verze tohoto koeficientu pro reálná čísla a vektory je nazývána Tanimotův koeficient (také rozšířený Jaccardův koeficient):

   $T(V_{1,}V_2)=\frac{V_1*V_2}{(|V_1|*|V_2|-V_1*V_2)}$ , kde $V_1$ a $V_2$ jsou vektory s reálnými čísly.

3. Diceův koeficient (Diceova míra) – je využívána hlavně ve statistice jako korelační koeficient mezi dvěma diskrétními událostmi. Pro statistické účely je definován Diceův koeficient pro dvě diskrétní události *A* a *B* jako: $D(A,B)=\frac{2*(P(\mathrm{A,B}))}{(P(A)+P(B))}$

   *P(A)* (*P(B)*) je pravděpodobnost, s jakou nastane samostatně událost *A* (*B*)
   *P(A,B)* je pravděpodobnost, s jakou události *A* a *B* nastanou současně
   Analogicky lze použít tento koeficient i pro vektory $V_1$ a $V_2$ $D(V_{1,}V_2)=\frac{2*(V_1*V_2)}{(|V_1|+|V_2|)}$

Podobným způsobem lze takto dále pokračovat např. pro porovnání dvou množin jako u Jaccardovy míry.

### 3.2.2 Shrnutí

Pro výběr jsem si zvolil implementovat vektorovou metodu. Myslím, že splňuje kompromis mezi náročnosti implementace, náročností na zdroje a přesností. Bayesův klasifikátor je jednoduchý a má i dostatečnou přesnost, ale pří rozšíření na multikategoriální klasifikátor rapidně vzrostou jeho nároky na zdroje.

Neuronové sítě jsou také velmi přesné, ale náročnost implementace přesahuje rámec bakalářské práce.

# 4 Implementace

Projekt je implementován v C/C++ za pomocí GUI frameworku Qt. Projekt je rozdělen do dvou částí. První část je mód, ve kterém program sbírá data. Druhá část je implementace vektorového modelu.

Proces učení začíná analýzou vstupních dat. Tato data mohou být tří typů:

1) soubor .csts (tedy výsledný soubor morfologické analýzy) slouží k natrénování především simulátoru morfologické analýzy

2) neformátovaný text + "*obor*" – z takovýchto dokumentů jsou vybírány termíny reprezentující daný obor

3) soubor .csts a "*obor*" – toto spojení zajistí kombinaci obou typů učení

Zde jsou data načítána a ukládána do slovníku. Pokud je vstupem soubor morfologické analýzy, je rozdělení na slova snadné. Neformátovaný text musí projí před vložením do slovníku lexikální analýzou a simulace lemmatizace.

V procesu analýzy je přijímán neformátovaný text. První dvě části, lexikální analýza a simulace lemmatizace, odpovídají fázi učení. V dalším kroku jsou však pro termíny vyhodnoceny váhy, které slouží jako vstup vektorovému modelu.

## 4.1 Lexikální analýza

### 4.1.1 Specifikace lexikální analýzy plain text dokumentů

Tuto část jsem řešil způsobem, který je rozebrán v teoretické části (víz 3.1.1). Zaměřil jsem se na dělení podle mezer. Výrazy spojené spojovníkem zpracovávám jako jedno slovo, protože slova spojená spojovníkem mohou mít trochu jiný význam než slova, která by vznikla rozdělením takovéhoto výrazu. V takového situaci lze předpokládat, že výraz bude chovat jako termín. Dle Luhna (Luhn, H. P. 1958) je předpoklad, že tento výraz bude identifikován jako sousloví. Dále jsem se rozhodl číslice nezpracovávat. Rozhodnutí podpořil fakt, že samotné číslice nemají schopnost rozlišit dokument. Často by se vyskytovaly na posledních místech, číslice se nechovají jako termín (Luhn, H. P.,1958).

Dalším krokem je identifikace fráze. Protože jde o sběr dvouslovných termínů, vycházím z předpokladu, že složením dvou gramatických slov, tedy slov bez sémantického významu, nebo spojením sémantického a gramatického slova, nelze vytvořit plnohodnotný termín. Jedná

se hlavně o případy, kdy vedle sebe stojí dvě spojky, dvě předložky nebo spojka a podstatné jméno. Zároveň vyřazuji i termíny se stejným lemmatem.

### 4.1.2 Implementace lexikální analýzy plain text dokumentů

Lexikální analýzu zajišťuje v programu třída *TextReader*, která načte řádek a ten dělí do vět. Základní rozdělení podle slovních mezer je zajištěno pomocí třídy ze **s**tandardní knihovny STL *std::stringstream*. Takto získané úseky se dále testují na přítomnost všech relevantních nealfanumerických znaků (např. ‘!’, ‘?’, ‘.’, ’,’,’:’, ‘” ”’) a podle přítomnosti jednotlivých znaků dochází k úpravě.

Protože nepoužívám přímo metody lemmatizace, ale snažím se je simulovat (viz 4.2.1), jsou některé části lexikální analýzy vloženy až za samotnou lemmatizaci. Jedná se například o tvorbu sousloví nebo o aplikaci negativního slovníku. Negativní slovník je simulován zjištěním, zda termín nepatří k zakázaným slovním druhům.

## 4.2 Lemmatizace

### 4.2.1 Postup lemmatizace

Vybral jsem si metodu simulující lemmatizaci se slovníkem. Hlavním důvodem byla absence nástroje pro morfologickou analýzu, který by byl použitelný pro Windows. Pro tuto simulaci je vytvářen **redundantní B-strom**, ve kterém probíhá hledání. Hledání probíhá ve dvou fázích.

V první fázi probíhá porovnání, zda analyzovaný tvar není zároveň slovníkový. Ve druhé fázi probíhá porovnávání podle tzv. **Levenshteinovy** metriky. Slovníkové tvary, které nejsou příliš vzdálené Levenshteinově metrice od hledaného výrazu, jsou považovány za vyhovující a je prozkoumáno celé slovo ze slovníku. Nejdřív jsou prozkoumány tvary. Pokud není hledaný výraz nalezen v tvarech slova, proběhne výpočet na porovnání celého načteného slova a hledaného tvaru. Porovnání je opět zajištěno pomocí Levenshteinovy metriky. Pokud vzdálenost vzdálenost nepřekračuje nastavenou mez (#define ANALYZ_LIMIT 0,25), je připraveno jako kandidát. Proces vyhledávání kandidátů končí ve chvíli, kdy jsou vybíraná slova příliš nepravděpodobná. Po prozkoumání všech možností se z nalezených kandidátů vybere slovo, ke kterému bude analyzovaný tvar přiřazen. (více v 4.6).

Při tomto typu vyhledávání jsou největším problémem slova nepravidelná (být - jsem,

nejsem), jejichž tvar je příliš odlišný od lemmatu. Pro tvary s takovouto charakteristikou jsou v B-stromě vytvářeny pomocné (zástupné) záznamy, které ukazují na lemma.

Úkolem nebylo vytvořit metodu pro morfologickou analýzu, která by byla schopna konkurovat statistickému taggingu. Tento modul vznikl, aby pro slova příbuzná existovala možnost nějakého zástupného slova, které by reprezentovalo výskyt v celé databázi. Proto byl zvolen tento postup. Je implementován jako jednoduchý modul, takže jej lze nahradit za jinou lemmatizační metodu. Morfologické značky jsou v programu zatím využívány jen minimálně, ale je to základ pro využívání sofistikovanějšího výběru termínů na základě jeho hodnot morfologických kategorií nebo spolupráce s dalšími lingvistickými nástroji.

### 4.2.2 Implementace lemmatizace

Redundantní B-strom je vytvářen ze souborů slovníku. Primárně se do B-stromu vkládá lemma bez doplňujících informací. Sekundárně jsou vkládány do B-stromu i tvary, které jsou v **Levenshteinově metrice** příliš vzdálené od lemmatu. Měření této vzdálenosti od lemmatu zajišťuje třída *AproximCompare*. Tvorba **B-stromu** je koncipována pouze pro jedno vytvoření, nikoliv pro úpravy.

Na začátku je vytvořen **obousměrný spojový seznam**, který obsahuje dva typy položek. Každá položka tohoto seznamu obsahuje **klíč** (lemma či tvar), ukazatele na levého i pravého souseda. Podle poslední položky se rozlišují na **záznam (*BRecordEnd*) a zástupce (*BRecordEndNeq*).**

Hlavní záznam ukazuje na začátek slova v souboru, proto jakákoliv úprava B-stromu by předpokládala změnu některého ze souborů slovníku. Takováto změna by však mohla změnit některé, v horším případě všechny, ukazatele do tohoto souboru. To je poměrně náročné hlavně na čas, uvážíme-li, že by tato operace byla brána jako základní. Proto jsem se rozhodl tuto operaci nahradit jednorázovým updatem celého slovníku (viz 4.7.2) po určitém počtu slov a poté smazat a znovu postavit celý **B-strom**.

## 4.3 Vážení

Metoda, kterou používám (viz 4.5), je často spojována s použitím tzv. *tfxidf*. Jedná se o základní váhovou funkci, jejíž princip ukrývá sám název. Váha termínu je rovna součinu *frekvence termínu x inverzní frekvence dokument*. Za frekvenci termínu lze použít jak

absolutní, tak relativní frekvenci. Za pojmem *inverzní frekvence dokumentů* se skrývá proměnná, která je rovna $ln\ (1/R_d)$, kde $R_d$ je relativní frekvence dokumentů, ve kterých se termín vyskytuje.

Výše popsaný výraz je brán jako základ váhové funkce, který se objevuje v několika variantách. Má úprava spočívá v penalizaci každého z termínů na základě určitých obecných úvah. Například *penalizace za výskyt ve více oborech* spočívá v testu, jak termín dokáže charakterizovat obor. Pokud má aproximativně stejný výskyt ve svých oborech, získává poměrně velkou penalizaci. Dále je prováděna penalizace v případě, kdy je tvar neotagovaný, ale v Levenshteinově metrice je dostatečně blízko, aby byl asociován s některým otagovaným nebo neotagovaným termínem. Pokud i termín, se kterým je vážený termín asociován, není otagovaný, je jeho penalizace větší.

Tento způsob vážení je v plném rozsahu využíván k získávání hodnot v databázi. Při získávání dat z jednoho dokumentu musí být funkce lehce pozměněna. V tomto případě odstraňuji složku *idf*. Tato složka je pro jeden dokument rovna 0 v případě, že složka výrazu $ln\ (1/R_d)$ $R_d = 1$. Pro získávání vah termínu v jediném dokumentu je využívána analogická funkce *tfxp*, kde jsou vybrané penalizace.

## 4.4 Extrakce termínů

Tato část byla jednou z nejtěžších na celé práci. Musí být vybrány takové termíny, které by nebyly ani příliš obecné, ani příliš vzácné.

Základ úvahy, kterou používám, spočívá v křivce tzv. **Zipfova zákona.** Zipfův zákon zjednodušeně říká, že součin frekvence slova a jeho pořadí (myšleno sestupně) lze aproximativně považovat za konstantu (více 2.10). Z toho lehce usoudíme, jak bude přibližně vypadat graf takové funkce (viz obr. 1). Dále využívám **Luhnovu úvahu,** která předpokládá existenci horní a dolní hranice pro graf, který je určen Zipfovým zákonem.

Slova nacházející se mezi horní a dolní hranicí jsou pak považována za významová slova pro daný dokument – termíny.

Nevýhodou těchto hranic je vágnost jejich definice, každý dokument je má jinak "nastavené". Jediná možnost, jak můžeme tyto hranice pro každý obor nastavit, je metodou pokusů a omylů.

## 4.5 Učení

### 4.5.1 Učení na otagovaných souborech

V prvním běhu programu (bez slovníku) jde o pouhou transformaci načítaných dat z otagovaného souboru. Při dalším spuštění učebního módu se předpokládá, že proběhlo učení také na plain textu (viz níže) a slovník může obsahovat nějaká neotagovaná data. V tomto případě, i když je tvar otagovaný, ho nechám vyhledat ve slovníku. Pokud tento tvar ve slovníku existuje, může dojít ke třem variantám.

První varianta nastane, pokud termín není otagovaný. U neotagovaných termínů přecházejí veškeré výskyty pod nový termín. Tento termín má stejné lemma jako nově přidávané slovo.

Další možností je, že je otagovaný termín i tvar. V tomto případě záleží, zda se tagy shodují nebo neshodují. Pokud se shodují, je práce u konce. Pokud se neshodují, je vyhledávané slovo uloženo do slovníku.

Nejtěžší varianta přichází v případě kdy otagovaný tvar lze najít ve slovníku, ale neotagovaný a navíc se tag termínu neshoduje s novým otagovaným tvarem. V tomto případě lze předpokládat, že se původní tvar spojil s termínem na základě blízkosti v Levenshteinově metrice. Je nutno přesunout již nasbíraná data z termínu ve slovníku do nově vznikajícího termínu.

### 4.5.2 Učení plain text

Termíny získané pomocí extrakce termínů jsou začleňovány do databáze společně s jejich hodnotami vah. Pokud je termín ve slovníku a má alespoň jeden tvar otagovaný, lze data lehce přidat.

Tvary, které nejsou nalezeny alespoň přibližně, jsou ukládány zvlášť a testují se, zda by nemohly patřit pod jedno lemma. Pokud podobnost dvou takovýchto tvarů je dostatečně vysoká, jsou sjednoceny v jeden termín. Lemma tohoto termínu je tvořeno nejdelším společným podřetězcem. Pokud je tento postup použit pro tři a více tvarů, existuje možnost, že vzdálenost některého z tvarů přeroste nastavené meze. Pokud bych při stavbě slovníku nepoužíval podružné záznamy ***(BRecordNeqEnd***) ve stromě (viz 4.2.2), byly by takovéto záznamy nevyhledatelné.

## 4.6 Vektorový model

Jak už bylo naznačeno, základní princip vektorového modelu je velmi jednoduchý (viz 3.2.1.3). Samotný výpočet zajišťuje jediná třída my_vector. K výpočtu je zapotřebí tří různých hodnot. První známe a lze ji zjistit už v módu učení, kdy rozšiřujeme slovní zásobu pro jednotlivé obory. Při načítání ji zajišťuje třída branch_info. Třída my_vector zajišťuje správný výpočet dalších dvou důležitých hodnot shromažďováním informací z jednotlivých termínů v dokumentu. Každý termín je načten a pro obory, ve kterých se termín vyskytuje, je spočtena váha termínu. Tato váha je uložena do instance třídy simpleV. Po skončení načítání termínů je v každé instanci simpleV jmenovatel ze základního vzorce[7]. Čitatel tohoto vzorce je tvořen součinem dvou hodnot. První je již zmiňovaná hodnota z třídy branch_info, která reprezentuje součet součinů všech vah termínů z oboru. Druhou hodnotou je hodnota součtu součinů vah termínů z testovaného dokumentu.

Třída my_vector je držitelem kompletních informací ze všech oborů. Získávání a čtení výsledků zajišťuje třída b_analyzator, která je rozšířením třídy my_vector. B-analyzator zajišťuje načítání dat z analyzovaných souborů a zároveň dat z databáze. Také zajišťuje předání výsledků.

## 4.7 Implementační poznámky – stručný popis tříd a algoritmů

### 4.7.1 Implementace třídy reprezentující slovo

Třída reprezentující slovo je spolu s třídou slovnik základními kameny celého programu. V programu existuje několik variant třídy, která drží informace o slově. Nejmenší z takovýchto tříd je třída *ref_word*. Tato třída obsahuje pouze lemma a ukazatel na pozici v souboru slovníku. Je využívaná hlavně při tvorbě záznamů pro B-stromu.

*Base_word* je třída, která je využívána hlavně při určování jednotlivých tvarů slov. Spolu s hodnotou, která reprezentuje pravděpodobnost, s jakou analyzovaný tvar spadá právě pod toto lemma. Base_word je zároveň přímým předchůdcem třídy *analyzed_word*. Tato třída v sobě spojuje výsledky analýzy jednotlivých tvarů. Tyto výsledky postupují dále k analýze, která je rozlišena podle módu, ve kterém zrovna program pracuje.

Pokud jde o učení, jsou odhadnuty hranice pro oddělení nevýznamových slov (viz Luhn) a z určených slov se stávají termíny, které jsou zaneseny do slovníku. V tomto bodě jsou z nich

7 viz 3.2.1.3 Vektororový model – kosinová míra

instance *learning_word*, které obsahují informace o oborech a zároveň na sobě mají množinu souborů, ve kterých se slovo vyskytuje. Po skončení učení jsou data ze všech souborů spojena pomocí merge algoritmu. Při tvorbě hlavního souboru se z instance *learning_wordu* stává instance *dict_word,* ve kterém je množina souborů nahrazena jejich absolutním i relativním počtem.

Při analýze dokumentu je z databáze načtena instance *dict_wordu (*pokud je tvar ve slovníku nalezen). Slovo projde váhovou funkcí a poznamená se aktuální váha v souboru a zjištěné váhy na trénovacích datech. Tato data poté přijímá implementace vektorového modelu *B-analyzator*.

### 4.7.2 Slovník

Tato třída je nejrozsáhlejší z celého programu a zajišťuje téměř všechny úkoly spojené s učením. Dokáže zpracovat jak soubor morfologické analýzy, tak plain-text soubor. Také zajišťuje konzistenci databáze.

Práci se soubory morfologické analýzy zajišťuje metoda slovníku *procces_tagged_file(),* jejímž argumentem je třída *data_file*, která reprezentuje soubor a předává důležité informace mezi jednotlivými částmi programu. Díky parseru z programu Morče (Český morfologický tagger), který mi byl laskavě zapůjčen k užívání mým vedoucím bakalářské práce, lze otagované soubory snadno číst a lze je poměrně snadno transformovat do požadované podoby.

Problémy mohou nastat s přidáním otagovaného tvaru, který už v databázi existuje jako neotagovaný. V tomto případě je nutné rozlišit, zda jde o termín, který je v databázi plnohodnotný (tedy alespoň jeden jeho tvar je řádně otagován), nebo zda vznikl při učení na plain textu (viz níže). V prvním případě jsou vyjmuta data pro termín a založen nový termín. V druhém případě je celému slovu přidáno lemma a tvar je otagován.

O něco málo komplikovanější je učení na neformátovaném (plain) textu. V tomto případě hledáme v databázi příslušný tvar. Pokud ho nenalezneme, pokusíme se alespoň aproximativně nalézt nějaké možnosti. Pokud ho nalezneme, je vrácen termín i s pravděpodobností, s jakou příslušný tvar spadá pod tento termín.

Pokud slovo není nalezeno, je pro něj založen nový termín. Jako dočasné lemma takového termínu je zprvu zvolen jediný tvar ve slově. Tvar je ovšem bez tagu. Pokud je nalezen jiný tvar, který je dostatečně blízko (myšleno v Levenshteinově metrice), je lemma termínu, které

vznikne spojením těchto dvou tvarů, jejich největší podřetězec (část, kterou mají společnou).

### 4.7.3 Reprezentace vlastního slovníku – B-strom

Implementace B-stromu je velice jednoduchá. Jedná se o klasickou datovou strukturu redundantní strom. Každý uzel (instance *BNode*) má 20 záznamů. Záznamy mohou být tří typů:

1) *BRecordEnd* obsahuje lemma a ukazatel do souboru, kde lze načíst celý termín
2) *BRecordEndNeq* obsahuje tvar slova a ukazatel na BRecordEnd
3) *BRecordIn* obsahuje ukazatel na BNode. Jeho klíč je kombinací řetězců z levého a pravého podstromu.

Tvorba B-stromu je složena ze dvou kroků. V prvním kroku se z hlavních souborů slovníku pomocí třídy *Bstore* vytvoří lineární spojový seznam instancí třídy *BrecordEnd* nebo *BRecordEndNeq*. Ve druhém kroku je tento spojový seznam rozdělen na uzly a je vytvořena poslední (listová) vrstva. Nad dvěma uzly je vytvořen BRecordIn s klíčem, který je tvořen kombinací klíčů maximálního prvku z levého podstromu a minimálního prvku z pravého podstromu.

Oproti klasickému B-stromu je ovšem ochuzen o funkce insert a delete pro jednotlivé záznamy. Toto zjednodušení vychází z předpokladu, že pokud se bude vkládat záznam do stromu, znamená to, že byl některý soubor updatován. Tím se však změnily všechny ukazatele, proto probíhá oprava slovníku jako smazání a znovu postavení B-stromu.

### 4.7.4 Zpracování souborů XML

Jedná se o soubory s jednoduchou strukturou. Každý soubor začíná kořenovým uzlem Slovnik, s výjimkou souboru se souslovími. Zde je kořenový uzlem Phrases.

Zápis a čtení těchto souborů zajišťují mnou vytvořené XML *readery* a *writery*. V některých případech je nutno spojit obě části a vytvořit třídu, která načítá soubor a zároveň ho vypisuje s určitými změnami v datech. Takové třídy vznikly pro přepis souborů slovníku nebo souborů se souslovím.

Implementace těchto rozsáhlých tříd, které takto spojují data, stojí na šablonovaci třídě base_rewriter. Tato třída očekává tři šablonovací parametry. Třídu pro zápis (např. *dictWord_writer, phrase_writer*), třídu pro čtení (*dictWord_reader, phrase_reader_doc*) a

třídu, která reprezentuje načítaná a zapisovaná data (*dict_word*,*doc_phrase*, ).

### 4.7.5 Zpracování souborů CSTS

Zpracování souboru morfologické analýzy je zajištěno pomocí kódu, který mi byl zapůjčen z projektu Morče (Český morfologický tagger) mého vedoucího bakalářské práce Mgr. Jana Raaba.

### 4.7.6 Kompilace a testování projektu

Ke kompilaci je nutné nainstalovat framework Qt.

Program byl vyvíjen a testován v operačním systému Win XP za pomoci nástrojů Microsoft Visual Studio a Qt creator (nástroj gcc 3.5.2 z MinGW).

# 5 Uživatelská příručka

Program byl navrhnut tak, aby analyzoval textové soubory na základě dat ze slovníku. Pro základ slovníku je vyžadován alespoň jeden otagovaný soubor. Proto je při prvním spouštění podobně jako při vytváření nového slovníku potřeba nejprve vycházet ze souboru morfologické analýzy. Tato data musíme nejprve uložit a poté znovu načíst. Nyní je možné se začít učit na trénovací množině textových dat.

## 5.1 Příkazy pro textový režim

Přehled argumentů:

[-{f/l}] [-{A/D}] [-{M/T}] *seznam souborů* {-B *seznam obor*ů} {d *folder*} {-s *file*} {-e}

-f / l -f – program očekává soubory pro přímé zpracování

-l – zadaný soubor je seznamem zpracovávaných souborů

-A/D – rozlišení práce se souborem (-y)

-A – analýza souborů

-D – soubory se zpracují jako součást trénovacích dat

-M/T – rozlišuje, jaký typ souborů je třeba očekávat

-M – soubor(-y) morfologické analýzy

-T – plain textová data

*soubory* – seznam jmen souborů, které se budou podle výše napsaných argumentů zpracovávat

– pozn. lze napsat plně kvalifikované jméno nebo relativní adresu

-B *obory* – seznam oborů, do kterých lze soubor(-y) zařadit

– pozn. tento argument je povinný při učení na plain textu

-d *folder* – složka obsahující slovník (v případě, že složka neobsahuje slovník a aktuální zpracovávané soubory mají podobu morfologické analýzy, je v *folder*

vytvořen nový slovník)

-s *file* – výsledky programu budou vypsány do XML souboru *file*

-*e* – program ukončí práci

Dodatečné argumenty:

**-h** – vypsání nápovědy

-backup *folder* – překopíruje slovník do složky backup v zadané složce

– předpokládá se, že zadaná složka obsahuje soubory slovníku

## 5.2 Popis GUI a popis práce v grafickém režimu

GUI je velmi jednoduché a intuitivní. Všechny ovládací prvky se nacházejí v horní liště pod záložkou File.

První položka Dictionary se vztahuje k načítání a ukládání slovníku:

Make new dictionary – vybereme nebo vytvoříme novou pracovní složku, ve které bude uložen slovník i dočasné soubory

Set dictionary – vybereme složku, ve které se nachází již vytvořený slovník

Load dictionary – načteme slovník z adresy udané pomocí položky Set dictionary (defaultně nastaveno na složku programu)

Unload dictionary – vymazání aktuálního slovníku z paměti

Save dictionary – uloží data získaná z učení

Další dvě položky lze použít, pokud je načten slovník.

Položka File(s) for learning slouží k přidání nových učebních souborů. Poslední položka File(s) for analyz slouží pro výběr souborů, které je potřeba zanalyzovat. Obě tyto položky rozlišují soubory pomocí podpoložek:

MorpAnalyz (csts) – pro výběr souborů morfologické analýzy

PlainText(txt) – pro výběr čistě plain textových souborů

Práce s tímto prostředím je poměrně jednoduchá. Při prvním spouštění opět vytvoříme slovník pomocí tlačítka Make new dictionary. Pokud už máme slovník vytvořen, najdeme ho

pomocí Set dictionary. Těmito tlačítky vybíráme celou pracovní složku slovníku - vlastní slovník je uložen v před připravené podsložce *data*. Pomocí tlačítka Load Dictionary načteme slovník do paměti a nyní vybíráme soubory pro analýzu nebo učení.

Po skončení práce se souborem je možné v levé části hlavního okna vidět seznam souborů. Označením některého ze souborů se v pravé části okna objeví zpracované údaje. Pro uložení nově naučených dat slouží položka Save Dictionary. Pokud tuto položku stisknete, upraví se soubory slovníku a celý se znovu načte.

Ve spodní části jsou čtyři tlačítka, která ovládají tato okna:

Save all results – zapíše všechny výsledky do XML souboru

Save current result – zapíše aktuální soubor do XML souboru

Delete current result – smaže vybraný soubor ze seznamu

Close without saving – zavře výsledkovou část a vyprázdní seznam výsledků

## 5.3 Interpretace výsledků

Jak při učení, tak při analýze jsou jedním z výsledků textové informace, které říkají, jak byl zadaný soubor zpracován. Při učení je výsledkem také změna souborů slovníku. Tedy v případě přidání nového oboru nebo zvýšení váhy slova v aktuálně přidávaném oboru (ukázka takto ukládaného slova příloha 4). V případě souboru morfologické analýzy jde o přidání nových tvarů a lemmat. Pokud tento text také patří do nějakého oboru, přidávají se i frekvence vybraných termínů a dvojic. V případě textového souboru jde pouze o vybrané termíny a dvojice.

Základní informací o souboru jsou počty slov, které jsou načteny a dále zpracovávány. První informací je tedy počet slov v celém souboru, dále pak počet zpracovaných slov a dvojic. Tento počet udává, kolik termínů a dvojic bylo ze souboru použito. (V případě, že zpracovávaný soubor je pouze obohacení slovníku novými otagovanými slovy, jsou tyto hodnoty na nule.)

Dalším údajem je čas, který byl na analýzu spotřebován. Je počítán od načtení po ukončení práce se slovníkem. Není zde započítáváno nahrávání slovníku do paměti a ani jeho úprava či destrukce.

Poslední informací o souboru jsou přiřazené obory. V učícím modu jde pouze o výpis zadaných souborů, při analýze jsou tyto informace vytvářeny z dosud naučeného slovníku. V současném nastavení se zobrazují 3 nejpravděpodobnější obory. (Pro výsledky 5.4.2 - je

chápano správné určení jako obor s největší pravděpodobností)

Všechny tyto informace se, ale ztratí pokud explicitně nejsou nastaveny parametry pro ukládání. Ukázka uložených výsledků v příloze 5

## 5.4 Výsledky

Články, které byly při testování použity, pochází z internetového archivu časopisu 21. století. Články byly vybírány náhodně dle kategorizace v tomto časopise. Pokud byl článek méně rozsáhlý, spojil jsem dva, nebo i tři články do jednoho souboru.

Testování proběhlo ve dvou krocích. V prvním kroku jsem testoval, jak úspěšný bude program na neotagovaných datech, a ve druhém, jak úspěšný bude program na otagovaných datech.

Trénovací množina dokumentů obsahovala texty z těchto oborů: technika, medicína, příroda a vesmír. Velikost množiny oborů závisí jen a pouze na vůli uživatele.

Jako výchozí hodnotu pro porovnání (baseline) byla uvažována úspěšnost 0,25 (25%).Tato baseline odpovídá situaci, kdy by bylo pouze tipováno do jakého oboru konkrétní dokument patří.

### 5.4.1 Učení

Tab.1 Popis průběhu učení z neotagovaných dokumentů

| Obor | Zpracovaných dokumentů | Celkový počet slov | Celkový počet zpracovaných lemmat | Celkový čas učení |
|---|---|---|---|---|
| technika | 39 | 68304 | 615 | 00:17:00 |
| medicina | 38 | 68527 | 722 | 00:19:07 |
| příroda | 51 | 112560 | 573 | 00:27:16 |
| vesmír | 36 | 55337 | 667 | 00:15:10 |

Tab.2 Popis průběhu učení z otagovaných dokumentů

| Obor | Zpracovaných dokumentů | Celkový počet slov | Celkový počet zpracovaných lemmat | Celkový čas učení |
|---|---|---|---|---|
| technika | 39 | 80644 | 9217 | 00:07:10 |
| medicina | 36 | 83315 | 8377 | 00:09:13 |
| příroda | 55 | 139940 | 14384 | 00:13:12 |
| vesmír | 25 | 55337 | 4569 | 00:03:28 |

### 5.4.2 Analýza

Testovací balíček I – náhodný výběr z učebních textů

Testovací balíček II – výběr zcela neznámých textů z časopisu 21. století

(balíčky I a II testovány na slovníku, který byl vytvořen z neotagovaných dat)

Testovací balíček III – výběr z otagovaných učebních textů

Testovací balíček IV – výběr z otagovaných neznámých textů

(balíčky III a IV testovány na slovníku, který byl vytvořen z otagovaných dat)

Tab.3 Porovnání úspěšnosti

| Testovací sada | Počet dokumentů | Úspěšnost | Počet slov | Průměrný počet slov na dokument | Celková doba zpracování | Průměrná doba zpracování jednoho souboru |
|---|---|---|---|---|---|---|
| Balíček I | 54 | 94,55% | 99275 | 1838.43 | 00:33:01 | 00:00:36 |
| Balíček II | 61 | 63,93% | 133912 | 2195.28 | 00:42:18 | 00:00:41 |
| Balíček III | 48 | 95,83% | 95217 | 1983.69 | 00:08:56 | 00:00:11 |
| Balíček IV | 55 | 81,82% | 134538 | 2446.14 | 00:10:55 | 00:00:11 |

V tabulce 3 je použita jedna z nejednodušších metrik, kdy porovnáváme počet dobře klasifikovaných dokumentů ku všem testovaným. Tato metoda však nezohledňuje počty v jednotlivých třídách.

V tabulce 4 je použita jedna z nejklasičtějších metrik pro porovnávání klasifikátorů je tzv.

$F_1$- $F_1=(2*P*R)/(P+R)$ míra. Jde o kombinaci tzv. Precision (P, přesnost)[8] a Recall (R, odezva)[9], kde :

$$P=D/(B+D)$$
$$R=D/(C+D)$$

Prvky B, C, D jsou prvky tzv. matice záměn a jsou definovány takto:

A – prvky, které byly klasifikovány jako negativní a jsou negativní

B – prvky, které označeny za pozitivní, ale měly být odmítnuty (negativní)

C – prvky, které byly odmítnuty, ale měly být klasifikovány jako pozitivní

D – prvky, které byly správně určeny jako pozitivní

Pro naše účely pozměníme definici:

B1 – jsou prvky, které byly zařazeny do špatné kategorie

C1– prvky, které měly být kategorizovány a nejsou

F1 mírou budeme tedy hodnotit pouze balíčky II, IV, zároveň také balíček V, který vznikl z otagovaných souborů použitých v balíčku II. Tyto soubory byly testovány na slovníku, který vznikl za použití neotagovaných souborů.

Tab.4 Porovnání úspěšnosti v míře $F_1$

| Balíček | D-prvky | B1-prvky | C1-prvky | Precision | Recall | F1 |
|---|---|---|---|---|---|---|
| Balíček II | 39 | 22 | 0 | 0,63 | 1 | 0,78 |
| Balíček IV | 45 | 10 | 0 | 0,81 | 1 | 0,9 |
| Balíček V | 39 | 21 | 0 | 0,67 | 1 | 0.78 |

### 5.4.3 Interpretace dosažených výsledků

U jednotlivých oborů lze předpokládat nějakou provázanost, díky této provázanosti můžeme očekávat jisté procento chyb.

Slovník jsem postupně naučil na 164 článcích od každého oboru. Soubor detailně popisující tento úkon lze na jít na přiloženém CD ve složce Učení. Tabulka A ukazuje pouze stručný výtah. Tabulka 3 obsahuje výsledky analýzy 115 neotagovaných a 105 otagovaných souborů a úspěšnost celé aplikace a rychlost celé aplikace.

8 míru Precision lze chápat jako pravděpodobnost s jakou klasifikovaný prvek spadá do oné konkrétní kategorie
9 míru Recall lze chápat jako pravděpodobnost s jakou může odmítnutý prvek spadat do testované kategorie

Aplikace podle očekávání vykazuje největší úspěšnost na datech, která byla použita pro vytvoření slovníků (z otagovaných i neotagovaných dat). Nejnižší úspěšnost pak vykazuje v případě, kdy je slovník z neotagovaných dat použit k analýze otagovaného souboru (baliček B2). Tento jev lze vysvětlit tím, že má pouhá simulace morfologické analýzy je poměrně neúspěšná proti statistickému taggingu.

Z výsledků vyplývá, že pokud jsou použity k učení i k analýze otagované soubory, je aplikace několikanásobně rychlejší a zároveň je i přesnější. Ovšem v prvních měřeních tomu tak nebylo. Rozdíl byl v jedné konstantě LIMIT_COS. Tato konstanta měla nastavenou příliš vysokou hodnotu, která vycházela z představy úhlu mezi dvěma vektory[10]. Praxe však ukázala, že především pro otagované soubory, bude termínů více než dostatek[11]. Hodnota, která byla důležitá pro porovnání se zmenšila asi o dva řády, ale přesto v porovnání s ostatními neztratila vypovídající hodnotu.

10 Při hodnotách blízkých nule jsou tyto dva vektory kolmé a přeneseně to znamená, že nemají nic společného.
11 U neotagovaného souboru je dost termínů vyloučeno proto, že nemají tag. Nepatří tedy s jistotou pod dané slovo.

# 6 Závěr

Cílem práce bylo vytvořit rychlý nástroj pro analýzu textů na základě reprezentativní množiny slov. Hlavním kritériem pro výběr slov je jejich počet v textu, ze kterého je na základě modelu *tfxidf* vytvořena váha slova. Jako porovnávací algoritmus je použit vektorový model s kosinovou mírou jako porovnávacím kritériem.

Díky absenci nástroje pro morfologickou analýzu pod Windows jsem byl nucen přidat modul, který morfologickou analýzu simuluje. Ke klasickému vyhledávání jsem přidal ještě aproximativní vyhledávání založené na Levenshteinově metrice, které je podle všechno poměrně úspěšné.

Výsledkem je nástroj, který rozlišuje data s přesností přes 70 % (myšleno v míře $F_1$) , ale jeho rychlost dosud neodpovídá praktickým podmínkám - zpracovaných 10 000 - 20 000 slov za minutu. Ke spodní hranici se blížíme při zpracování tagovaných souborů (podle tabulky 2 a 3). Pro praktické použití by byla nutná optimalizace, např. zlepšit spolupráci objektů nebo provádět ve větší míře vláknově také samotné určování. Dalším rozšířením by mohla být lepší spolupráce s nástrojem morfologické analýzy. Ke zlepšení úspěšnosti kategorizace by mohla přispět kombinace několika měr založených na vektorovém modelu nebo více využívat tagů ve slovech.

V původním návrhu jsem uvažoval i o ručně tagovaných slovech. Během práce se tato vlastnost ukázala být téměř zbytečnou. V množství několika tisíc lemmat nemá jedno slovo až takový význam.

# Použitá literatura

21. století [online]. Praha: RF HOBBY, s. r. o, 2003-2010 [cit. 2010-05-20]. Dostupné z WWW: <http://www.21stoleti.cz>.

Bachmanová, Jarmila, et al. Encyklopedický slovník češtiny. 1.vydání. Praha: NLN, 2002. 604 s. ISBN 80-7106-484-X.

Forsberg, Marcus ; Wilhemsson, Kenneth. Project: Automatic Text Classification [online]. 1998 [cit. 2010-05-20]. Automatic Text Classification with Bayesian Learning. Dostupné z WWW: <http://www.cse.chalmers.se/alumni/markus/LangClass/LangClass.pdf>.

Garcia, E. . Mi Islita [online]. 2005 [cit. 2010-05-20]. Dostupné z WWW: <http://www.miislita.com/>.

Hadi, Wa`el Musa; Thabtah, Fadi; Abdel-Jaber, Hussein A Comparative Study using Vector Space Model with K-Nearest Neightboor on Text Categorization Data. In World Congress on Engineering 2007 [online]. Vol I. London: Newswood Limited, 2.-4. 7. 2007 [cit. 2010-05-20]. Dostupné z WWW: <http://www.iaeng.org/publication/WCE2007/WCE2007_pp296-300.pdf>. ISBN 978-988-98671-5-7.

Hajič,Jan: Statistické modelování a automatická analýza přirozeného jazyka (morfologie, syntax, překlad) 2001 Slovenčina a čeština v počítačovom spracovaní (zborník referátov zo seminára Bratislava 26.-27.10.2001 (ed.A. Jarošová))

Dostupné z www <http://ufal.mff.cuni.cz/publications/year2001/slovko1.doc>

Hajičová, Eva;Panevová Jarmila; Sgall, Petr. Úvod do teoretické a počítačové lingvistiky.

Praha: Karolinum, 2003. 156 s. ISBN 80-246-0470-1.

Hynek, Jiří; Ježek, Karel . Automatická klasifikace dokumentů do tříd za použití metody Itemsets. Datakon 2001 . 2001, 1, s. 1-8. Dostupný také z WWW: <http://textmining.zcu.cz/publications/Itemsets%20Datakon%202001.pdf>. ISSN 80-227-1597-2.

Janik, Maciej; Kochnut, Krys OmniCat: Automatic Text Classification with Dynamically Defined Categories. In 7th International Semantic Web Conference (ISWC2008) [online]. Karlsruhe: ISWC, 2008 [cit. 2010-05-20]. Dostupné z WWW: <http://data.semanticweb.org/conference/iswc/2008/paper/poster_demo/75/html>.

Joachims, Thorsten Text Categorization with Support Vector Machines: Learning with Many Relevant Features. In European Conference on Machine Learning (ECML) [online]. European Conference on Machine Learning (ECML):Springer, 1998 [cit. 2010-05-20]. Dostupné z WWW: <http://www.cs.cornell.edu/people/tj/publications/joachims_98a.pdf>.

Kolektiv autorů Ústavu českého jazyka FF Masarykovy univerzity v Brně. Příruční mluvnice češtiny. 2.vydání. Praha: NLN, 2003. 799 s. ISBN 80-7106-134-4.

Luhn, H.P. The Automatic Creation of Literature Abstracts. IBM Journal of Research and Development [online]. 1958, 2, 2, [cit. 2010-03-20].Dostupný z WWW: <http://ieeexplore.ieee.org/xpl/tocresult.jsp?isnumber=5392664>. ISSN 0018-8646.

Rohn, Jiří. Linearní algebra a optimalizace. Praha: Karolinum, 2004. Vektorové prostory, s. 199. ISBN 80-246-0932-0.

Ryjáček, Zdeněk ; Čada, Roman; Kaiser, T. Skripta Diskrétní matematika. Praha: Vydavatelstvím ZČU, 2004. Orientované grafy, s. 91-97. Dostupné z WWW: <http://www.cam.zcu.cz/~ryjacek/students/DMA/skripta/dma.pdf>.

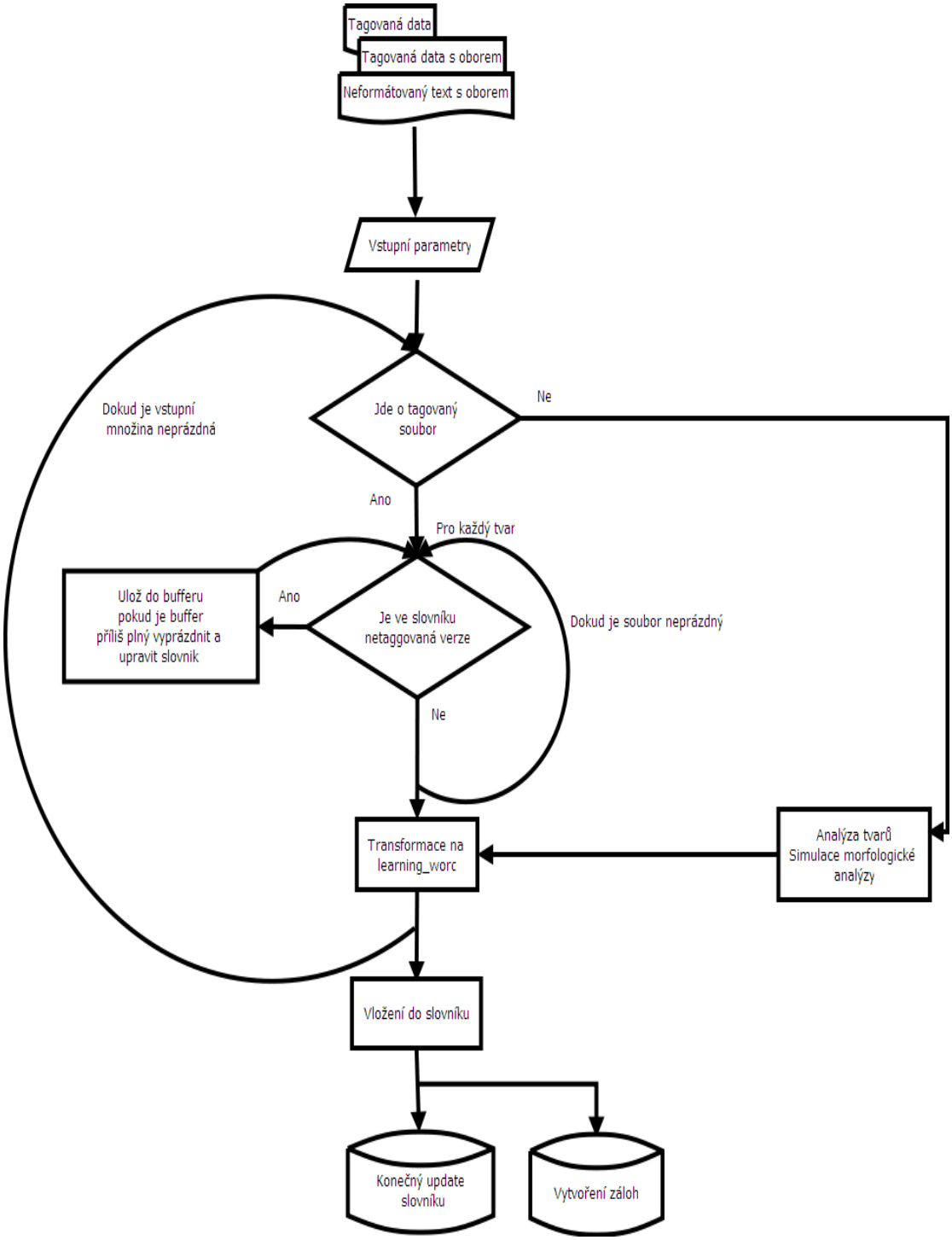
Schwarz, Jiří. Současný stav a trendy automatické indexace dokumentů [online]. 2.0. Praha: 2002, 2003 [cit. 2010-05-20]. Dostupné z WWW: <http://full.nkp.cz/nkdb/docs/studie/souh.html>.

Skopal, Tomáš . Neuronové síte a Information Retrieval. [online]. 2002, 1, [cit. 2010-05-20]. Dostupný z WWW: <http://www.cs.vsb.cz/arg/techreports/neural.pdf>.

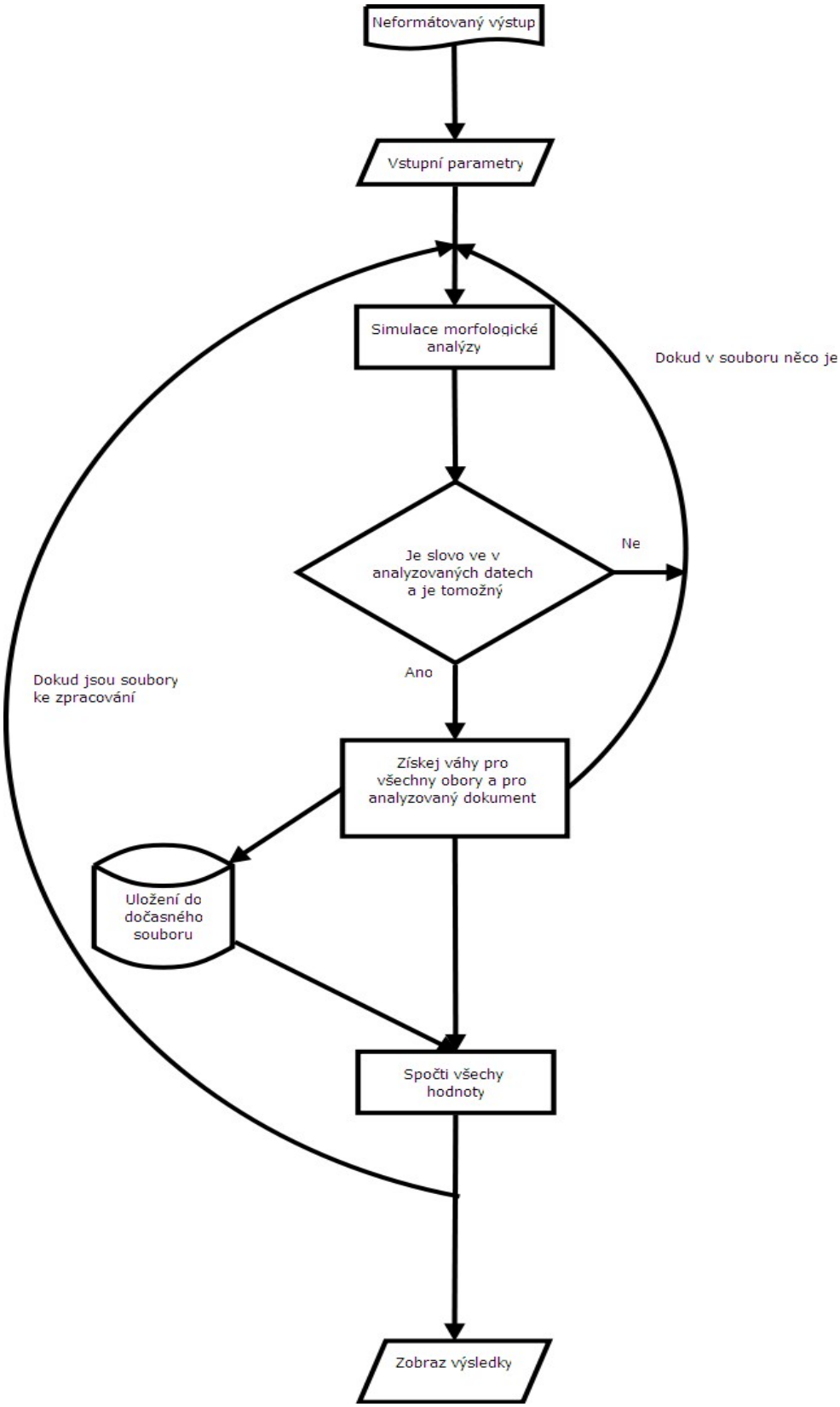
Staněk, Tomáš. Photon System Instrumentals [online]. [cit. 2010-05-20]. Užití Bayesovských metod ve strojovém učení. Dostupné z WWW: <http://www.psi.cz/ftp/sad0ur/statistika-iso.pdf>.

# Přílohy

Příloha 1 - Jednoduchý UML diagram průběhu učení



Příloha 2 - Jednoduchý UML digram pro průběh analýzy



Příloha 3 - Struktura DTD XML souboru s výsledky

```
<! ELEMENT name (#PCDATA) >
<! ATTLIST name                          // obsahuje celou cestu k zpracovávanému souboru
        typ  (learning | analyzed) >     // typ procesu jakým soubor prošel

<! ELEMENT pocet (#PCDATA) >             // počet zpracovaných slov
<! ATTLIST pocet
        processed_words  (#PCDATA)       // kolik slov bylo zpracováno jako termíny
        processed_phrase (#PCDATA)       // kolik dvojic bylo zpracováno
        up_cutoff  (#PCDATA)     // hodnota horního cut-off - limit při příliš vysoké frekvenci
        down_cutoff (#PCDATA) > // hodnota dolního cut-off - limit při příliš nízké frekvenci

<! ELEMENT time (#PCDATA) >              // čas zpracování souboru v milisekundách

<! ELEMENT branch (#PCDATA) >            // název oboru
<! ATTLIST branch
        probability  (#PCDATA) >         // pravděpodobnost, s jakou do oboru patří

<! ELEMENT branches (branch *) >         // záznamů branch může být 0-3

<! ELEMENT result (name,pocet,time,branches) >  // záznam pro jeden soubor
<!ELEMENT results (result *) >           // záznamů result může být 0-n
```

Příloha 4 - Ukázka uložení informací z analýzy souboru

```
<result>
  <name typ="analyzed">
F:/Data/Skola/Projects/Rocnikovyprojekt/Rprojekt/Rprojekt/Final/texty/textyearning/Balik_A/animal1.txt
</name>
<pocet processed_words="28" processed_phrase="935" up_cutoff="0.0282026"
cut_dow_cutoff="0.00158888">
      5035
    </pocet>
 <time>95203</time>
 <branches>
  <branch probability="22.09" >technika</branch>
  <branch probability="40.14" >medicína</branch>
  <branch probability="92.86" >příroda</branch>
</branches>
</result>
```

Příloha 5 - Ukázka z slova ze slovníku

```
<term>
 <lemma tag="" >hedvábí</lemma>
<shapes>
   <subterm pocet="8" probability="0" >
     <tvar>hedvábí</tvar>
     <tag>NNNS4-----A----</tag>
</subterm>
</shapes>
 <branches>
   <branch name="příroda" probability="0">
      <word pocet="8" relative="7.10732e-005" ></word>
      <doc pocet="1" relative="0.0196078" ></doc>
  </branch>
</branches>
<frekvency>
  <word pocet="8" relative="2.61671e-005" ></word>
  <doc pocet="1" relative="0.00609756" ></doc>
</frekvency>
</term>
```

Příloha 6 - Hlavní soubory a složky na CD

[AnalyzatorCD] :

Applications (složka obsahující bínární verzi Analyzátoru textu)

Dictionary (slovníky, které byly využívány pro testování)

SourceCode (zdrojové kódy pro překlad )

Results (soubory s průběhem učení a analýzy)

Texts (texty, které byly používané pro učení a analýzu)

Analyzator_textu.pdf (elektronická verze bakalářské práce)