VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

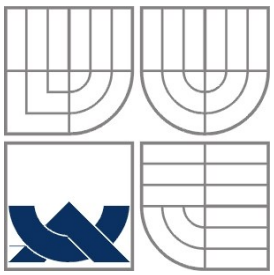
INTELIGENTNÍ EMAILOVÁ SCHRÁNKA

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

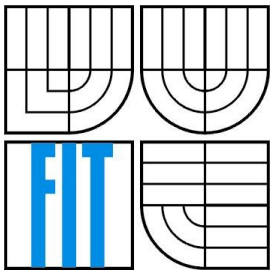
AUTOR PRÁCE
AUTHOR

Bc. ANTONÍN POHLÍDAL

BRNO 2012



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

INTELIGENTNÍ EMAILOVÁ SCHRÁNKA

INTELLIGENT MAILBOX

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

Bc. ANTONÍN POHLÍDAL

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. PETR CHMELAŘ

BRNO 2012

Vysoké učení technické v Brně - Fakulta informačních technologií

Ústav informačních systémů Akademický rok 2011/2012

Zadání diplomové práce

Řešitel: **Pohlídal Antonín, Bc.**
Obor: Informační systémy
Téma: **Inteligentní emailová schránka**
Intelligent Mailbox
Kategorie: Databáze

Pokyny:

1. Seznamte se s problematikou získávání znalostí z databází s důrazem na klasifikaci; nastudujte relevantní literaturu. Zaměřte se na klasifikaci textu.
2. Seznamte se s implementací IMAP serveru.
3. Po dohodě s vedoucím demonstrujte funkčnost existujících nástrojů užívaných pro klasifikaci nestrukturovaných dat na vzorku spamu.
4. Na základě experimentů si zvolte vhodnou metodu, tu implementujte. Vytvořte demonstrační systém, který bude klasifikovat příchozí poštu.
5. Zhodnoťte vlastnosti použitých nástrojů a přínos vytvořené aplikace, případně její rozšíření.

Literatura:

- HAN, J., KAMBER, M. *Data Mining: Concepts and Techniques*. Morgan Kaufmann Publishers, 2006, ISBN 1-55860-901-6.
- Feldman, Ronen. The text mining handbook: advanced approaches in analyzing data. New York: Cambridge University Press, 2007. 410 s., ISBN 0-521-83657-3.
- CHMELAŘ, P., STRYKA, L. *Multimediální databáze.* 2007.

Při obhajobě semestrální části diplomového projektu je požadováno:

- První dva body zadání.

Podrobné závazné pokyny pro vypracování diplomové práce naleznete na adrese http://www.fit.vutbr.cz/info/szz/

Technická zpráva diplomové práce musí obsahovat formulaci cíle, charakteristiku současného stavu, teoretická a odborná východiska řešených problémů a specifikaci etap, které byly vyřešeny v rámci ročníkového a semestrálního projektu (30 až 40% celkového rozsahu technické zprávy).

Student odevzdá v jednom výtisku technickou zprávu a v elektronické podobě zdrojový text technické zprávy, úplnou programovou dokumentaci a zdrojové texty programů. Informace v elektronické podobě budou uloženy na standardním nepřepisovatelném paměťovém médiu (CD-R, DVD-R, apod.), které bude vloženo do písemné zprávy tak, aby nemohlo dojít k jeho ztrátě při běžné manipulaci.

Vedoucí: **Chmelař Petr, Ing.**, UIFS FIT VUT
Datum zadání: 19. září 2011
Datum odevzdání: 23. května 2012

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
Fakulta informačních technologií
Ústav informačních systémů
612 66 Brno, Božetěchova 2

L.S.

doc. Dr. Ing. Dušan Kolář
vedoucí ústavu

Abstrakt

Tato diplomová práce se zabývá využitím klasifikace textu při třídění příchozí pošty. Nejdříve je popsána problematika získávání znalostí z databází a je detailně rozebrána klasifikace textu s popisem vybraných metod. Dále je uveden princip emailové komunikace a jsou popsány protokoly SMTP, POP3 a IMAP. Následuje návrh implementace systému, který klasifikuje příchozí poštu a rozbor použitých technologií, tedy Apache James Server, PostgreSQL a RapidMiner. Na závěr je uvedena implementace všech jednotlivých částí výsledného systému a jsou provedeny experimenty s testovací sadou emailů Enron Dataset.

Abstract

This master‘s thesis deals with the use of text classification for sorting of incoming emails. First, there is described the Knowledge Discovery in Databases and there is also analyzed in detail the text classification with selected methods. Further, this thesis describes the email communication and SMTP, POP3 and IMAP protocols. The next part contains design of the system that classifies incoming emails and there are also described realated technologie ie Apache James Server, PostgreSQL and RapidMiner. Further, there is described the implementation of all necessary components. The last part contains an experiments with email server using Enron Dataset.

Klíčová slova

získávání znalostí z databází, klasifikace textu, klasifikace emailů, naive bayes, logit boost, support vector machines, augmented latent semantic indexing spaces, radial basis function networks, email, pop3, imap, apache james server, rapidminer, postgresql, enron email dataset

Keywords

knowledge discovery in databases, text classification, email classification, naive bayes, logit boost, support vector machines, augmented latent semantic indexing spaces, radial basis function networks, email, pop3, imap, apache james server, rapidminer, postgresql, enron email dataset

Citace

Pohlídal Antonín: Inteligentní emailová schránka, diplomová práce, Brno, FIT VUT v Brně, 2012

Inteligentní emailová schránka

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně pod vedením Ing. Petra Chmelaře. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Antonín Pohlídal
22. 5. 2012

Poděkování

Chtěl bych poděkovat panu Ing. Petrovi Chmelařovi za jeho odborné vedení, rady a pomoc při tvorbě této diplomové práce. Poděkování patří také mé rodině za velkou podporu a neocenitelné rady.

Obsah

1 Úvod

Cílem této diplomové práce je nastínit problematiku získávání znalostí z databází se zaměřením na klasifikaci, konkrétně se jedná o klasifikaci textu. Důležitou součástí je rozbor problematiky a principů emailové komunikace s popisem komunikačních protokolů. Z těchto znalostí se poté vychází při návrhu výsledného systému pro klasifikaci emailů. Před samotnou implementací je proveden rozbor možností zvolených technologií. Při implementaci se vychází ze zvoleného emailového serveru Apache James Server, který je vhodně přizpůsoben, aby umožňoval klasifikovat příchozí emailové zprávy.

Toto téma považuji za velmi zajímavé, přínosné a aktuální, protože emailová komunikace je v dnešní době nedílnou součástí našeho života. Denně se lidé potýkají s problémem třídění příchozí pošty. Zpočátku se zdálo být jako vhodné řešení použití filtrovacích pravidel. Tento způsob je ale náročný na správu, nedokáže třídit neznámý typ příchozí pošty a s rostoucím počtem filtrovacích pravidel roste i chybovost při třídění zpráv. Z těchto důvodů mě proto zaujala možnost vytvoření emailové schránky, která by tyto problémy řešila.

Druhá kapitola obsahuje základní informace o historickém vývoji získávání znalostí z databází. Dále popisuje možnosti využití této techniky spolu s výhodami získávání nových znalostí. Následně je popsán průběh celého procesu získávání znalostí a je uvedena obecná architektura dolovacího systému. Na závěr jsou zmíněny možné druhy dat pro dolování a jsou dále rozebrány typy dolovacích úloh.

Ve třetí kapitole je uveden popis jednotlivých fází klasifikace a vybraných klasifikačních metod. Jedná se Naive Bayes, LogitBoost, Support vector machines, Augmented latent semantic indexing spaces a Radial basis function networks. Poté je popsána základní architektura systému pro dolování z textu. Na závěr jsou uvedeny možnosti zpracování dat, jedná se o extrakci klíčových slov, odstraňování stop-slov a princip váhovací metody TF-IDF.

Kapitola čtvrtá uvádí historický vývoj emailové komunikace a motivaci pro používání emailu. Dále je uveden celý proces přenosu emailové zprávy a její struktury, tedy obsahu hlavičky a těla zprávy. Na závěr je popsán přenosový protokol SMTP a komunikační protokoly POP3 a IMAP, které jsou také porovnány.

Obsahem páté kapitoly je návrh výsledného systému. Je zde nejprve rozebrán princip klasifikace emailů a popsán výběr dolovacího nástroje RapidMiner. Dále jsou uvedeny metody textové klasifikace RapidMineru. Následuje rozbor výběru emailového server Apache James Server a databázového systému PostgreSQL. Na závěr je shrnut a popsán princip spolupráce.

Šestá kapitola obsahuje rozbor použitých technologií. Je zde tedy popsán základní princip fungování emailového serveru Apache James Server, databázového systému PostgreSQL a dolovacího nástroje RapidMiner.

V sedmé kapitole je nejprve uveden výsledný diagram komponent. Poté následuje popis způsobu implementace klasifikační podpory pro emailový server Apache James Server a provedené úpravy databázového schématu. Na závěr je rozebrán princip tvorby klasifikačního modelu a jeho následné aplikace na vstupní emaily.

Kapitola osmá uvádí nejprve informace o testovací množině emailů Enron Dataset a následně jsou s touto sadou provedeny experimenty. Jedná se například o porovnání klasifikačních metod Support vector machines nebo vliv dat na úspěšnost klasifikace.

Poslední kapitola obsahuje závěr, ve kterém je provedeno zhodnocení celé této práce a jsou navrženy různé možnosti vylepšení a rozšíření do budoucna.

2 Získávání znalostí z databází

Na úvod této kapitoly bych rád uvedl výstižnou definici podle [1]: „Můžeme říci, že *získávání znalostí z databází* je extrakce (neboli „dolování“) zajímavých (netriviálních, skrytých, dříve neznámých a potencionálně užitečných) modelů dat a vzorů z velkých objemů dat. Tyto modely a vzory reprezentují znalosti získané z dat.“

Uvedená definice vystihuje samotnou podstatu získávání znalostí, kde je důležité, že se jedná o skryté a netriviální informace, které nejsou na první pohled zřejmé. Za získávání znalostí tedy v žádném případě nemůžeme považovat provádění jednoduchých výpisů z databáze, protože z takto získaných informací se nedozvíme vůbec nic nového.

Tato kapitola je velmi důležitá pro tuto práci, protože jsou v ní popsány základní principy, ze kterých vycházím v praktické části. Stěžejním prvkem je právě celý proces získávání znalostí z databází popsaný v kapitole 2.3. Hlavního principu tohoto procesu je využito při tvorbě výsledného procesu klasifikace emailů, který je detailně popsán a rozebrán v kapitole 7.4.

2.1 Vývoj

Získávání znalostí z databází je výsledkem postupného a hlavně přirozeného vývoje databázové technologie. Z tohoto důvodu [1] rozděluje vývoj získávání znalostí na několik důležitých období podle toho, jak docházelo k rozvoji databázové technologie.

V 60. letech minulého století začaly vznikat datově intenzivní aplikace na podporu činnosti firem a jiných organizací. Vzrůstajícímu objemu dat nevyhovovala současná podpora pro práci se soubory. Vyvstal proto požadavek na lepší podporu ukládání perzistentních dat a jejich další správy. Z těchto důvodů bývá toto období označováno jako období vzniku databázové technologie. Jako první vznikly hierarchické a síťové databáze a k nim odpovídající systémy řízení báze dat. Roku 1970 publikoval zaměstnanec firmy IBM Corporation Edgar Frank Codd základy teorie relačního modelu dat, na nichž byly vybudovány relační databáze a odpovídající systémy řízení báze dat.

Relační systémy zaznamenaly konečné vítězství na trhu v 80. letech minulého století. Došlo tedy k výraznému rozvoji metod a nástrojů na podporu návrhu relačních databází. S tím souvisí začátek výskytu pokročilých modelů dat, které se snažily rozšířit relační model nebo poskytnout odlišný model dat podporující složitě strukturovaná data nebo specifický způsob práce s daty. Jedná se například o objektově orientovaný model dat nebo model pro deduktivní databáze. Dále se také objevují aplikačně orientované systémy řízení báze dat pro prostorové, temporální, vědecké databáze, na podporu inženýrských aplikací, CAD systémů apod. V tomto období tedy dochází k velkému rozvoji databázové technologie a současně s tím dochází k výraznému nárůstu objemu dat uložených v databázích. Vlivem všech těchto faktorů je vyvíjen tlak na dostupnost technologií a nástrojů pro analýzu velkého množství dat

V 90. letech minulého století se tedy objevují technologie a nástroje pro budování datových skladů obsahujících data zejména z produkčních databází. Datové sklady umožňují data získaná z různých datových zdrojů shromažďovat a připravovat pro následnou analýzu. Problémem je, že produkční databáze sice poskytují podporu pro operativní řízení v podobě různých tiskových sestav a přehledů, ale jejich hlavním účelem je zajištění databázových transakcí. Tato činnost se označuje zkratkou OLTP (Online Transaction Processing). Provádět analýzy nad produkčními daty je neúnosné jednak z hlediska zátěže, ale také proto, že pro analýzu můžeme potřebovat historická data, která naopak nepotřebujeme v produkční databázi. Datové sklady tedy oddělují prostředí pro provádění

analýz od produkční databáze a poskytují podporu pro ukládání dat a samotnou analýzu. Model dat zde má podobu vícerozměrné datové kostky, nad kterou jsou definovány operace. Tato oddělená analýza s operacemi se označuje jako OLAP (Online Analytical Processing).

Dále se [2] zmiňuje, že kromě analýzy podporované OLAP operacemi se v 90. letech minulého století ve vědeckých kruzích začíná mluvit o získávání znalostí z databází. První podnět přišel z Ameriky, kde se na konferencích věnovaných umělé inteligenci pořádaly první workshopy věnované této problematice. Na rozdíl od OLAP operací, které se používají interaktivně, je jádrem procesu získávání znalostí automatizované nalezení potencionálně užitečných modelů a vzorů v datech. Tento proces je blíže popsán v kapitole 2.3.

Během 90. let minulého století se dále objevují nové typy dat a ty je potřeba ukládat v databázích. Jedná se například o multimediální data a různě strukturované dokumenty. Dále dochází k rozvoji Internetu a služeb, které poskytuje, například WWW (World Wide Web). Na web můžeme pohlížet jako na velkou distribuovanou heterogenní databázi obsahující nejrůznější typy dat uložených v souborech nebo databázích.

V období po roce 2000 nastává rozvoj metod a nástrojů používaných pro získávání znalostí a především jejich použití v praxi. Dochází také k dalšímu výzkumu v oblasti databázových technologii a získávání znalostí. Jedná se například o tzv. proudy dat, což jsou data, která vznikají souvisle v nějakém zdroji. Tato data je potřeba spravovat a analyzovat v reálném čase. Většinou se tedy nedá počkat a analýzu provést později, protože by data byla již zastaralá.

2.2 Motivace

Jak již bylo popsáno v kapitole 2.1, je vznik a vývoj získávání znalostí úzce spjat s rozvojem databázové technologie. Zpočátku databáze obsahovaly jen ta nejnutnější data a proto ani nebylo potřeba provádět složitější analýzy. Postupem času ovšem dochází k vývoji hardwaru i softwaru a tím se ukládání dat stává levnější a dostupnější. Množství informací v databázích prudce roste, ale hlavně se zvětšuje rozdílnost uložených dat. Lidé si začínají klást otázku, zda je možné získat nějaké nové znalosti z již existujících informací. Právě tato vidina nových znalostí, které nikdo jiný nemá, je velkou motivací a hnací silou rozvoje a použití získávání znalostí z databází. Velké databáze přímo lákaly k propojení a k provádění nejrůznějších analýz a vyhodnocení. Získávání znalostí z databází si získalo velkou pozornost a oblibu, protože umožňuje přeměnit data na užitečné znalosti.

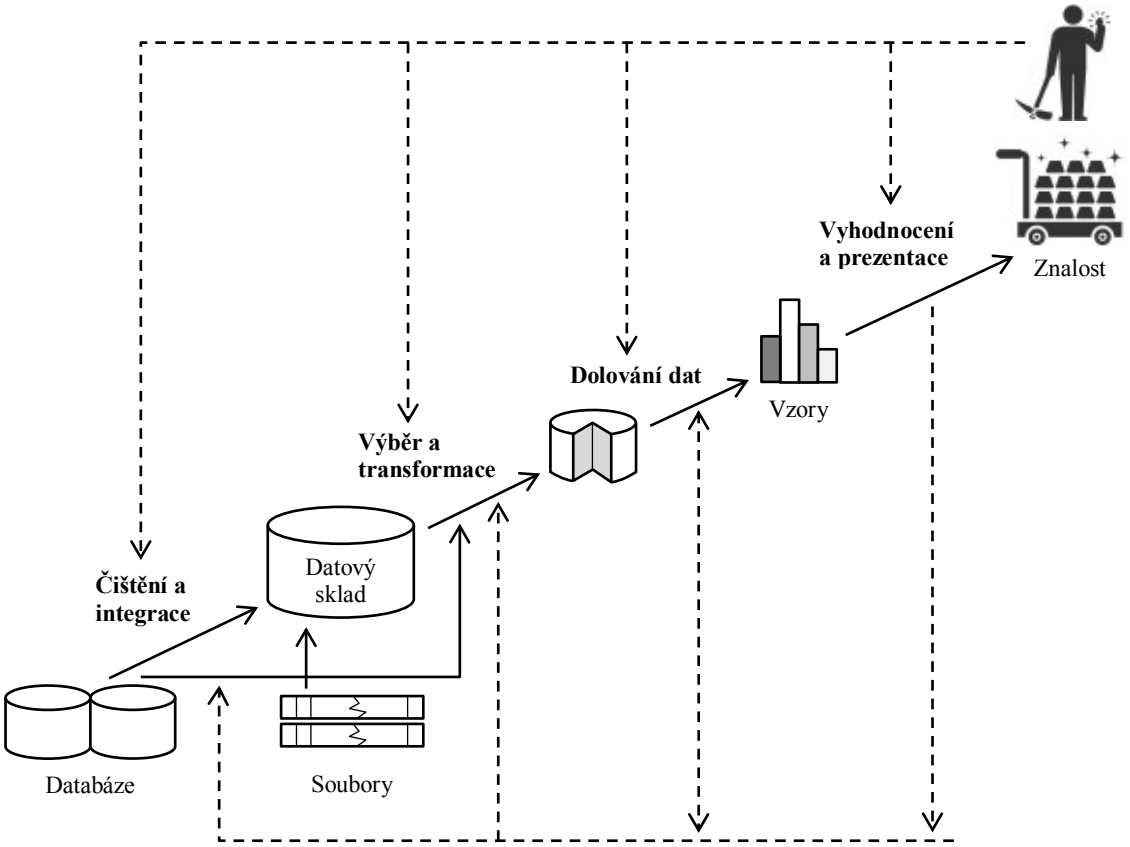
Získávání znalostí z databází se využívá pro řízení a rozhodování ve velkém množství nejrůznějších oborů lidské činnosti. Jedná se například o analýzu trhu, tedy rozbor prodeje a poptávky, přilákání nových a udržení stávajících zákazníků, řízení státní správy a podniků nebo detekce podvodů. V posledních letech se zvětšuje poptávka po analýze proudu dat. Příkladem jsou data z kamerových systémů, telefonní hovory, informace o provozu v počítačové síti nebo analýza městského provozu. Mnohokrát si ani neuvědomíme, že jsme narazili na použití získávání znalostí. Internetové obchody dnes již běžně uvádí informace o tom, jaké produkty si uživatelé nejčastěji kupují společně s vybraným zbožím. Toto je možné právě díky analýze a následnému vyhodnocení předešlých nákupů.

2.3 Proces získávání znalostí

Získávání znalostí z databází je ve své podstatě proces skládající se z několika po sobě jdoucích krocích, které se zpravidla v určitých cyklech opakují. Podle [1] a [3] se jedná o těchto sedm kroků:

1. **Čištění dat** – v této první fázi je nutné se vypořádat s chybějícími daty, provést odstranění vzniklého šumu, ale také vyřešit možné nekonzistence dat.
2. **Integrace dat** – provádí se spojování dat, protože vstupní data mohou pocházet z různých zdrojů. Čištění a integrace se často provádí společně jako jedna fáze souhrnně nazývaná předzpracování. Výsledná data jsou následně uložena například do datového skladu.
3. **Výběr dat** – z datového úložiště se provádí výběr jen těch dat, která jsou relevantní a vhodná pro řešení zadané analytické úlohy.
4. **Transformace dat** – cílem je transformovat nebo konsolidovat data do vhodné podoby pro dolování. Toto se provede například vykonáním sumarizačních nebo agregačních operací. Občas se transformace a konsolidace dat provádí ještě před fází výběru dat, zejména když se jedná o datový sklad, protože může být součástí tvorby datového skladu.
5. **Dolování dat** – jedná se o jádro celého procesu získávání znalostí. Je to proces, ve kterém se aplikují určité metody za účelem získání vzorů z dat, respektive vytvoření modelu dat.
6. **Hodnocení vzorů** – v této fázi je potřeba identifikovat skutečně zajímavé vzory, které reprezentují znalosti založené na míře užitečnosti.
7. **Prezentace znalostí** – získané znalosti se na závěr zobrazí uživateli za pomoci nejrůznějších technik vizualizace a reprezentace znalostí.

První čtyři kroky tohoto procesu provádí různé formy úprav dat, a proto se dají shrnout jako přípravná fáze. Jedná se tedy o předzpracování dat, při kterém probíhá příprava na dolování. Obrázek 2.1, který je uveden níže, přehledně zobrazuje celý popsaný proces získávání znalostí.



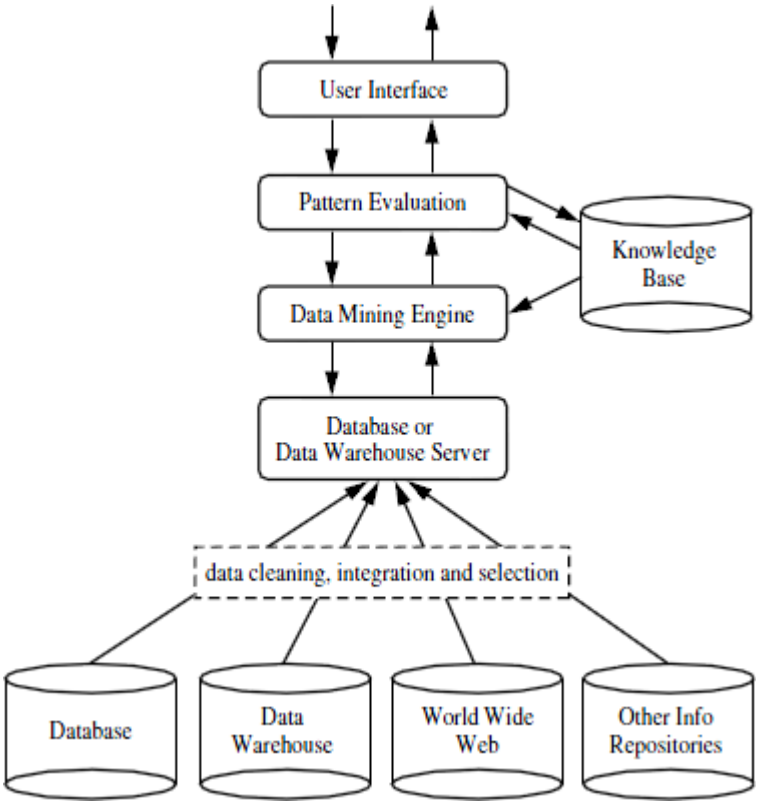
Obrázek 2.1 Proces získávání znalostí (převzato z [3])

2.4 Architektura dolovacího systému

Předchozí kapitola 2.3 obsahovala detailní popis celého procesu získávání znalostí. Jestliže z tohoto procesu vycházíme, potom [3] uvádí, že architektura typického dolovacího systému může obsahovat následující důležité komponenty:

- **Databáze, datový sklad, world wide web nebo další zdroje informací** – jedná se o jednu nebo o množinu databází, datových skladů, tabulek nebo dalších zdrojů dat. Nad daty se mohou provádět metody čištění a integrace.
- **Databáze nebo server datového skladu** – databáze nebo server datového skladu je odpovědný za načítání relevantních dat, založených na dolovacích požadavcích uživatele.
- **Báze znalostí** – jedná se o znalosti používané při vyhledávání nebo vyhodnocování zajímavých vzorů. Mohou zahrnovat například uživatelské požadavky, které mohou být použity například pro posouzení, zda je zkoumaný vzor zajímavý nebo ne.
- **Dolovací modul** – je to nezbytná součást dolovacího systému skládající se z množiny funkčních modulů pro úkoly jako je charakterizace, asociativní a korelační analýza, klasifikace, predikce, shluková analýza, analýza odlehlých hodnot a evoluční analýza.
- **Modul pro vyhodnocení vzorů** – typicky tato komponenta využívá měření důležitosti a dále spolupracuje s dolovacím modulem, aby se provádělo hledání skutečně zajímavých vzorů
- **Uživatelské rozhraní** – jedná se o modul, který zprostředkovává komunikaci mezi uživatelem a dolovacím systémem a provádí prezentaci výsledků nebo vstupních dat.

Obrázek 2.2, uvedený níže, přehledně zobrazuje popsanou architekturu dolovacího systému.



Obrázek 2.2 Architektura dolovacího systému (převzato z [3])

2.5 Data pro dolování

Dle [1] můžeme v zásadě dolovat v jakémkoliv druhu dat. Jednak to mohou být perzistentní data, uložená v nějakých úložištích, ale také data transientní, jakými jsou proudy dat. Když se podíváme na způsob uložení dat podle typu databáze, potom je [4] rozlišuje na tyto:

- **Relační databáze** – sestává se z tabulek, relací sloupců a jejich hodnot, které jsou dostupné přes systém řízení báze dat.
- **Transakční databáze** – podle [1] se obecně jedná o soubor, ve kterém každý záznam reprezentuje jednu transakci. Transakce tedy typicky zahrnuje jednoznačný identifikátor transakce a seznam položek, které transakci tvoří. Jedná se například o obchodní transakce, jako je nákup zboží, nebo o bankovní transakce a další.
- **Datové sklady** – jsou to úložiště, která kombinují data z různých zdrojů. Dle [1] je obvykle datový sklad modelován jako multidimenzionální databázová struktura, tedy tzv. multidimenzionální datová kostka, kde každá dimenze odpovídá atributu nebo skupině atributů ve schématu a každá buňka obsahuje agregované údaje.
- **Další techniky** – k výše uvedeným variantám umožňují použití objektového přístupu nebo lze kombinovat jiné externí zdroje. Jedná se například o disková pole nebo Internet.

Dále [4] říká, že lze také rozlišovat dle uložených dat na tyto:

- **Temporální a časově uspořádaná data** – jedná se o klasické relační a transakční databáze, ve kterých převládají data strukturovaná, tedy text a číselné informace s časovými razítky.
- **Prostorová data** – vztahují se k určitému prostoru, který reprezentují. Tato data jsou uložena jako rastr nebo jako vektorová informace s popisem a lokalizací. Příkladem mohou být satelitní nebo medicínské snímky, ale i geografické mapy.
- **Multimediální databáze** – [5] říká, že multimediální systémy řízení báze dat poskytují podporu pro správu multimediálních dat jako doplněk tradičních databázových systémů. Mohou obsahovat složité texty, grafiku, obrazy, video fragmenty, mapy, hlasy, hudbu a další podoby audio-vizuálních informací.
- **Textové databáze** – obsahují obecně textové dokumenty, tedy například XML, HTML nebo další dokumenty s potencionálně různorodými informacemi.

2.6 Typy dolovacích úloh

„Systémy pro dolování dat zpravidla dávají možnost řešení různých typů dolovacích úloh, navíc často i různými algoritmy. Je to jednak proto, že musí být dostatečně univerzální, ale i proto, že uživatelé často nemají představu o tom, jaký model jejich dat může být užitečný.“ Takto charakterizuje [1] typy dolovacích úloh a také uvádí důvody proč je vlastně potřeba mít různé typy. Dále říká, že můžeme typy dolovacích úloh rozdělit do dvou základních skupin, tedy:

- **Deskriptivní** - tyto úlohy charakterizují obecné vlastnosti analyzovaných dat. Typickým příkladem je určení, které zboží zákazníci kupují společně. Tohoto docílíme, když provedeme analýzu nákupního košíku všech zákazníků.

- **Prediktivní** – ty provádí dedukci pro předpověď budoucího chování na základě analýzy současných dat. Do této kategorie patří například klasifikace, která je více popsána v následující kapitole 2.6.3 nebo v kapitole 3.

2.6.1 Popis konceptu/třídy

Jedná se o jeden ze základních typů dolovacích úloh. Podle [3] mohou být data asociována s třídou nebo konceptem. Mějme například prodejnu s elektronikou, ve které máme dvě třídy zboží a to počítače a tiskárny. Zákazníci se poté dají rozdělit na dva základní koncepty, jeden jako utrácivý a druhý jako šetřivý. Dále může být užitečné popsat individuální třídy a koncepty v souhrnné, stručné, ale stále přesné formě. Vzniklý popis se nazývá *popisem třídy/konceptu*. Tyto popisy můžeme získat buď charakterizací, nebo diskriminací dat.

Charakterizace dat podle [1] značí sumarizaci obecných vlastností analyzované třídy. Data, která odpovídají uživatelem specifikované třídě, jsou typicky získány za pomoci databázového dotazu. Existuje několik metod charakterizace a sumarizace dat. Řadí se k nim využití OLAP operace *roll-up* nad datovou kostkou, kterou lze využít k uživatelem řízené sumarizaci. To se provede tak, že vytvoříme abstraktnější pohled na data analyzované třídy. *Indukce orientovaná na atributy* je příkladem automatizované metody.

Diskriminace dat je [3] definována jako porovnání obecných vlastností cílové třídy, obsahující datové objekty, s obecnými vlastnostmi objektů z jedné nebo z množiny porovnávaných tříd. Cílové a porovnávané třídy mohou být specifikovány uživatelem a za pomoci databázového dotazu se získají příslušné datové objekty. Metody, které se používají pro diskriminaci dat, jsou podobné metodám pro charakterizaci dat.

2.6.2 Frekventované vzory

Jak již název napovídá, jedná se o vzory, které se často vyskytují v datech. Podle [3] existuje mnoho druhů opakujících se vzorů zahrnující frekventované množiny, sekvence a podstruktury. Pod frekventovanou množinou si můžeme představit takové položky, které se často objevují společně v určité množině prvků. Například nákup často obsahuje pečivo a ovoce nebo zeleninu. Frekventovaná sekvence znamená, že se daná (opakovaná) činnost provádí v určitých krocích po sobě. Příkladem může být nákup herní konzole a poté opakující se nákup nejrůznějších her. Podstruktury jsou například grafy, stromy nebo mřížky, které se mohou kombinovat s množinami a sekvencemi. Pří dolování frekventovaných vzorů získáváme zajímavé asociace a korelace.

Asociační analýza hledá, jaké vztahy existují mezi vstupními (analyzovanými) daty. Tyto vztahy se podle [3] vyjadřují asociačními pravidly, které nám znázorňují, jaké hodnoty atributů se vyskytují společně ve vstupní množině dat. Asociační pravidlo je tvaru

$$A \Rightarrow B, \text{ kde } A, B \text{ jsou množiny položek } \{i_1, i_2, \dots, i_m\}, \tag{2.1}$$

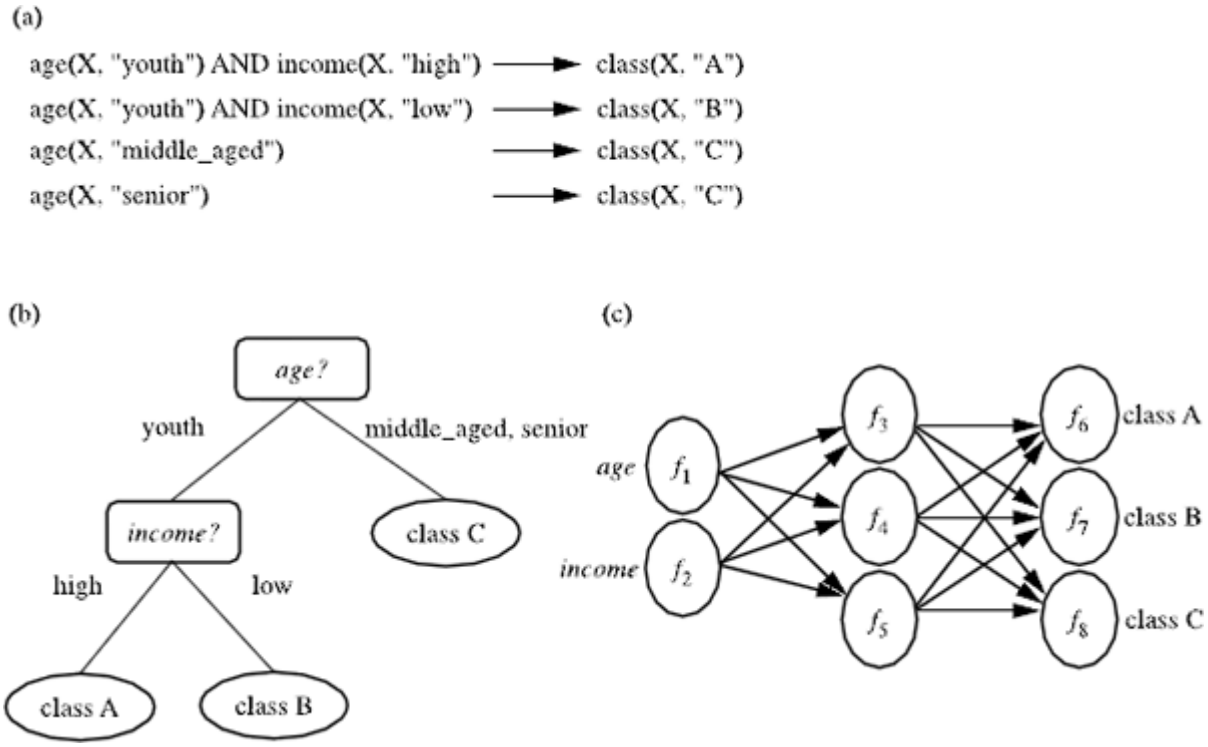
vyskytující se v transakci T. Jejich kolekce tvoří transakční databázi D. Pravidlo se pak zapisuje:

$$i_1 \wedge i_2 \wedge \dots \wedge i_k \Rightarrow i_{k+1} \wedge i_{k+2} \wedge \dots \wedge i_{k+l} \tag{2.2}$$

Číslo $k + l$ označuje celkovou kardinalitu (mohutnost) K asociačního pravidla.

2.6.3 Klasifikace a predikce

Klasifikace je podle [3] proces při němž se hledá model (nebo funkce), která nejlépe popisuje a současně rozlišuje třídy dat nebo koncepty. Potom se tento vytvořený model použije pro odhadnutí do jaké třídy zařadit objekt, jehož zařazení neznáme. Vytvořený model je založený na analýze množiny trénovacích dat, tedy na množině objektů, u kterých známe zařazení do tříd. Klasifikační model může mít podobu například klasifikačních pravidel (if-then), rozhodovacího stromu, matematické formule nebo neuronové sítě. Obrázek 2.3 zobrazuje vybrané příklady modelů.



Obrázek 2.3 Modely klasifikace (převzato z [3]): (a) if-then (b) rozhodovací strom (c) neuronová síť

Celý popsaný proces klasifikace se tedy dá shrnout do těchto tří fází:

1. **Trénování** – v této fázi se provádí analýza tzv. trénovacích dat a na základě této množiny se provede vytvoření základního modelu klasifikace pro dané třídy.
2. **Testování** – nyní do celého procesu vstupují testovací data, na kterých se provádí hodnocení kvality vytvořeného modelu. Správnost modelu je dána procentuální úspěšností klasifikace.
3. **Použití** – tato fáze již zahrnuje používání vytvořeného modelu, tedy na základě tohoto modelu se provádí klasifikace vstupních objektů, jejichž třídu neznáme.

Důležité tedy je, že u trénovacích a testovacích dat známe třídy pro příslušné objekty a také obě tyto množiny dat musí být disjunktní, tedy nesmí mít žádné společné objekty. Po fázi testování se na základě výsledků rozhodneme, jestli daný model vyhovuje nebo zda je potřeba ho vytvořit znovu, případně upravit jeho parametry.

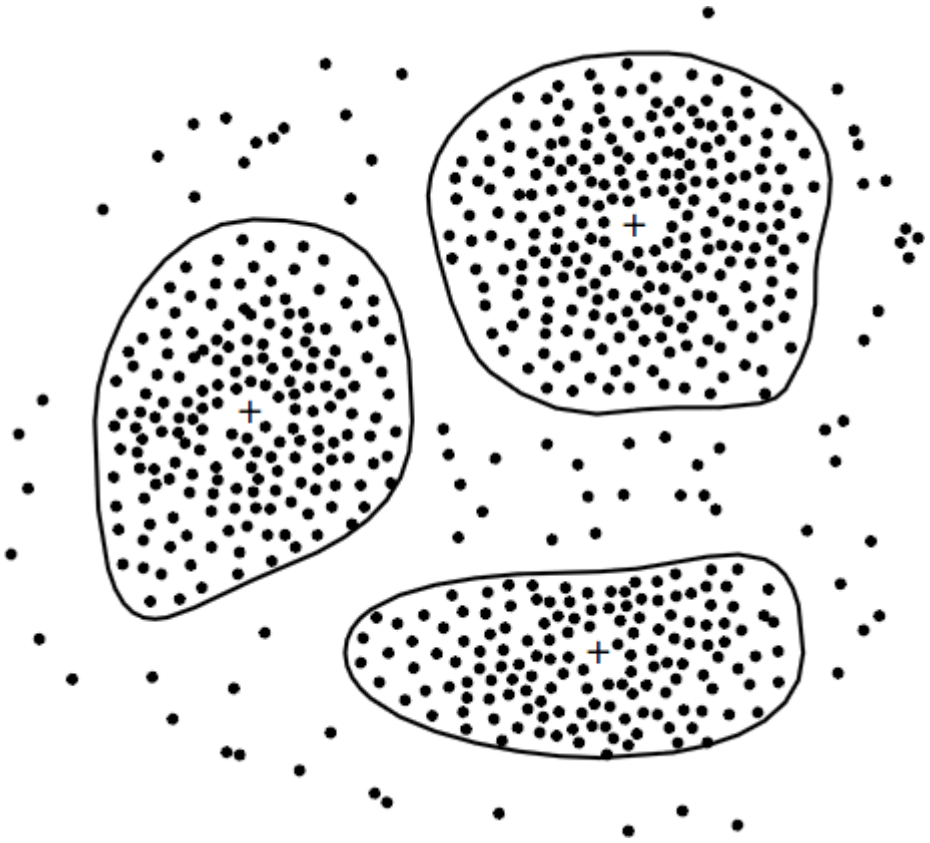
Ohledně predikce [1] říká, že: „Klasifikace se používá k predikci hodnot kategorických (diskrétních, neuspořádaných) hodnot tříd (například *dobrý_klient*, *rizikový_klient*). Pro hodnoty spojitých atributů (např. objem prodeje) používáme predikci. Znamená to, že v tomto případě predikujeme chybějící nebo nedostupnou *numerickou hodnotu*. Nejčastější metodou predikce je *regresní analýza*.“

2.6.4 Shluková analýza

Shluková analýza (anglicky nazývaná Cluster Analysis) je podle [3] proces seskupování daných objektů do shluků, nazývaných také třídy nebo clustery, ve kterých mají tyto objekty stejné nebo velmi podobné vlastnosti. Objekty v jednom shluku potom mají jiné nebo méně podobné vlastnosti, než objekty v dalším shluku.

Problém může nastat při určování, co náleží do jednoho shluku a co již patří do jiného. Třeba rozlišení zda se jedná o muže nebo ženu je daleko jednodušší, než se snažit zjistit, které ženy mají podobné způsoby při nakupování. Proto se při posuzování podobnosti dvou objektů berou v potaz všechny jejich vlastnosti, na základě kterých se často za pomocí vzdálenostní funkce vypočítá míra podobnosti těchto dvou objektů. Následně se tato míra využije při rozhodování, zda se objekt zařadí do nějakého shluku anebo se ukončí shlukování, protože si již žádné objekty nejsou podobné, aby se daly zařadit do shluku. Po ukončení shlukování se nám může stát, že některé shluky obsahují jen jeden objekt. Těmto objektům se říká odlehlé objekty (anglicky bývá nazývané Outliers) a jejich hledáním se zabývá analýza odlehlých objektů, která je více popsána v kapitole 2.6.5. Někdy nás totiž mohou zajímat právě tyto objekty, které se více odlišují.

Obrázek 2.4, který je uveden níže, zobrazuje ukázku shlukové analýzy. Jedná se o zákazníky obchodu *AllElectronics* vzhledem k jejich místu bydliště v daném městě. Analýza nad těmito zákazníky provede jejich rozdělení do homogenních sub-populací. Každá takto vzniklá skupina může reprezentovat cílovou skupinu při tvorbě například marketingové kampaně.



Obrázek 2.4 Ukázka shlukové analýzy (převzato z [3])

2.6.5 Analýza odlehlých hodnot

Jak již bylo popsáno v předešlé kapitole 2.6.4, může se nám stát, že po skončení shlukové analýzy některé shluky obsahují jen jeden objekt. Jedná se o odlehlé objekty a jejich hledáním se zabývá analýza odlehlých hodnot. V určitých situacích nás totiž mohou zajímat právě tyto odlišné objekty. Jak uvádí [3], většina dolovacích metod tyto odlehlé objekty odstraňuje jako šum v datech. Avšak v některých případech může být důležité získat právě tyto odlehlé hodnoty. Jedná se například o odhalování podvodů nebo zjišťování podezřelého chování.

Dále Kamber říká, že odlehlé objekty mohou být odhaleny použitím statistických testů, které předpokládají rozdělení dat nebo pravděpodobnostní model. Dále se také používá měření vzdáleností, kde jsou jako odlehlé objekty identifikovány ty, které jsou značně vzdáleny od jakéhokoliv jiného shluku dat. Naopak metody založené na derivacích zjišťují odlehlost na základě rozdílnosti v hlavních charakteristikách dané skupiny objektů.

2.6.6 Analýza evoluce

Podle [1] analýza evoluce popisuje a modeluje pravidelnosti a trendy u objektů, jejichž chování se mění v čase. Dříve zmíněné metody lze použít na data, která mají nějaký vztah k času, ale přesto existují speciálně zaměřené metody, jako je analýza podobnosti nebo vyhledávání periodicity. Příkladem může být burza a investování. Jestliže máme k dispozici data z burzy za několik posledních let, potom je možné nad těmito daty provést dolování. Identifikujeme tak pravidelnosti například v celkovém vývoji cen akcií i akcií jednotlivých společností. Díky tomuto můžeme předvídat budoucí vývoj cen, což může přispět při rozhodování o rizikovosti investice.

2.7 Shrnutí

Na počátku této kapitoly je uvedena základní definice získávání znalostí z databází a je také zdůvodněno, proč je tento obor tak důležitý pro celou práci. Dále následuje stručné shrnutí celého historického vývoje získávání znalostí z databází v souvislosti s vývojem ukládání dat. Poté jsou rozepsány různé možnosti využití této techniky a je uvedena základní motivace k získávání nových znalostí z velkých kolekcí dat. Následuje podrobný popis celého průběhu procesu získávání znalostí z databází, což je demonstrováno na názorném obrázku. Na základě tohoto procesu je dále definována typická architektura dolovacího systému, což je základ pro praktickou část této práce. Následně jsou zmíněna data pro dolování jednak z pohledu typu databáze a poté vzhledem k typu uložených informací. Na závěr jsou rozebrány jednotlivé typy dolovacích úloh s uvedenými příklady.

3 Klasifikace textu

Účelem klasifikace textu je rozhodnout, do jaké třídy se má daný text zařadit. Může se jednat o jednoduché dělení na dvě třídy nebo o složitější, kde je definováno více různorodých tříd. Aby bylo možné klasifikovat určitý text, je potřeba provést získávání znalostí z textu. Jedná se o proces, který je ve své podstatě odvozen od získávání znalostí z databází popsaný v kapitole 2.

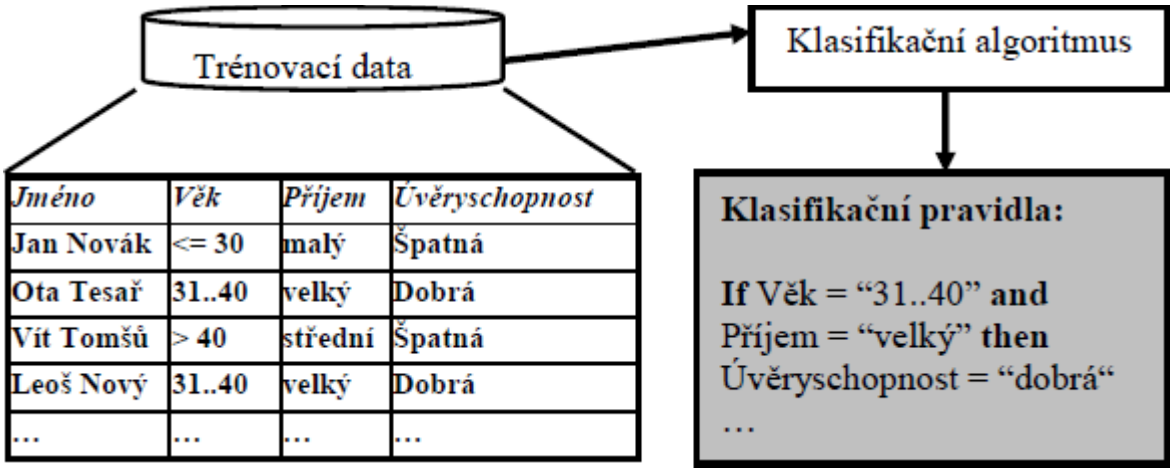
Podle [6] můžeme získávání znalostí z textu obecně definovat jako proces založený na znalostech, ve kterém uživatel pracuje s analytickými nástroji nad kolekcemi dokumentů. Podobně jako u získávání znalostí z databází se získávají užitečné informace z datových zdrojů na základě identifikace a zkoumání zajímavých vzorů. V tomto případě jsou za datový zdroj považovány kolekce dokumentů a zajímavé vzory se nehledají ve formalizovaných záznamech v databázi, ale v nestrukturovaných textových datech uložených v dokumentech, které tvoří kolekce.

Klasifikace textu je tedy proces, při němž se hledá model, který nejlépe popisuje a současně rozlišuje třídy daných dokumentů. Poté se vytvořený model použije pro odhadnutí do jaké třídy zařadit dokumenty, jejichž zařazení neznáme.

3.1 Fáze klasifikace

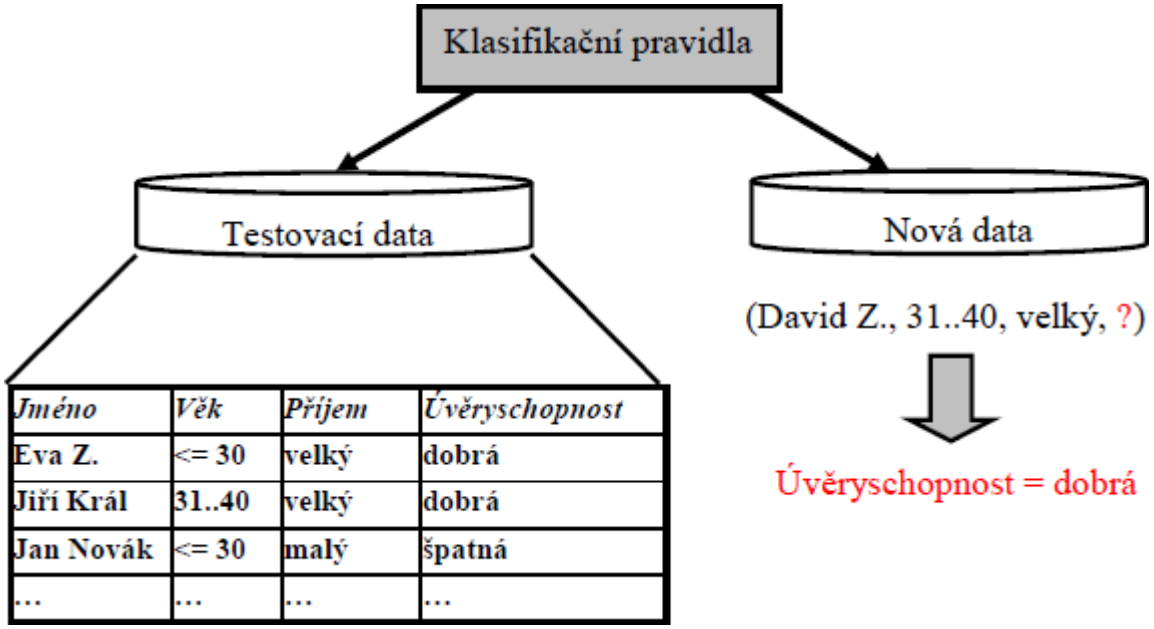
Jak bylo popsáno v úvodu této kapitoly, je potřeba nejprve vytvořit model s pomocí kterého se bude provádět klasifikace. Nyní si proto podle [1] uvedeme jednotlivé fáze klasifikace textu, které vedou k vytvoření výsledného modelu. Ten je ve své podstatě založený na analýze množiny trénovacích dat, tedy na množině dokumentů, u kterých známe zařazení do příslušných tříd. Celý zmíněný proces klasifikace se proto dá shrnout do těchto tří fází:

1. **Fáze trénování** – často bývá nazývána také jako fáze učení. V této fázi se provádí analýza trénovacích dat, což jsou vybrané vzorky dokumentů, u kterých předem známe jejich zařazení. Tento vzorek dat je vstupem pro klasifikátor, jehož úkolem je vytvořit tzv. klasifikační model. S pomocí tohoto modelu se s jistou přesností může klasifikovat vstupní dokument do dané třídy. Obrázek 3.1, který je uvedený níže, zobrazuje tuto trénovací fázi klasifikace.



Obrázek 3.1 Fáze trénování klasifikace (převzato z [1])

2. **Fáze testování** – druhá fáze zahrnuje testování vytvořeného klasifikátoru. Nyní do celého procesu vstupují testovací data, na kterých se provádí hodnocení kvality vytvořeného modelu. Opět se proto vybere vzorek dokumentů, u kterých je známa jejich klasifikační třída. Tyto vzorky ale musí být odlišné od trénovacích dat. Množina testovacích dokumentů musí tedy být tzv. disjunktní k množině trénovacích dokumentů. Vytvořený klasifikátor nyní provede rozřazení testovacích dokumentů do příslušných tříd. Díky tomu, že známe správné zařazení, je možné zjistit procentuální míru úspěšnosti. Na základě této úspěšnosti může být nyní rozhodnuto, zda daný model vyhovuje nebo jestli je potřeba ho vytvořit znovu, aby bylo dosaženo lepších výsledků. Obrázek 3.2, který je uvedený níže, zobrazuje popsanou fázi testování klasifikace.



Jméno	*Věk*	*Příjem*	*Úvěryschopnost*
Eva Z.	<= 30	velký	dobrá
Jiří Král	31..40	velký	dobrá
Jan Novák	<= 30	malý	špatná
…	…	…	…

Obrázek 3.2 Fáze testování klasifikace (převzato z [1])

3. **Fáze použití** – jedná se o závěrečnou fázi, která zahrnuje používání vytvořeného klasifikátoru. Pro jednotlivé vstupní dokumenty se provádí klasifikace, při které se uplatňuje vytvořený model. Během této fáze se může ovšem zjistit, že je potřeba upravit parametry klasifikátoru, případně vytvořit nový. To může být zapříčiněno například změnou požadavků. V tom případě se vracíme na začátek celého procesu, tedy do fáze trénování, abychom mohli vytvořit nový klasifikační model.

3.2 Metody klasifikace

Tato kapitola obsahuje popis základních principů vybraných klasifikačních metod. Nejprve je popsána metoda Naive Bayes, což je jednoduchá a hlavně rychlá metoda založená na výpočtu pravděpodobnosti. Poté je rozebrána metoda LogitBoost, která vytváří klasifikační model na základě předchozí zkušenosti. Další metoda Support vector machines provádí dělení prostoru na hyperroviny, čímž dostává oddělené třídy. Augmented latent semantic indexing spaces využívá stěžejních prvků pro určení příslušných kategorií. Na závěr je rozebrána metoda Radial basis function networks, která využívá strukturu tří vrstev pro vytvoření výsledného klasifikátoru.

3.2.1 Naive Bayes

Jedná se o algoritmus, který je podle [7] založen na teorii pravděpodobnosti a je odvozen z Bayesovy rozhodovací teorie. Pravděpodobnost, že zpráva d je ve třídě c se označuje $P(c|d)$ a spočítá se jako

$$P(c|d) \propto P(c) \prod_{k=1}^{m} P(t_k|c), \tag{3.1}$$

kde $P(t_k|c)$ je podmíněná pravděpodobnost termu t_k vyskytujícím se ve zprávě třídy c a $P(c)$ je předchozí pravděpodobnost zprávy vyskytující se ve třídě c.

V emailové klasifikaci je třída zprávy zjištěna tak, že se najde nejpravděpodobnější nebo maximální třída c_{MAP}. Ta je definována takto

$$c_{MAP} = arg \max_{c \in \{c_l, c_s\}} P(c|d) = arg \max_{c \in \{c_l, c_s\}} P(c) \prod_{k=1}^{m} P(t_k|c). \tag{3.2}$$

Vzhledem k tomu, že rovnice (3.2) zahrnuje násobení mnoha podmíněných pravděpodobností, pro každý term, může výpočet vést k podtečení. V praxi se násobení pravděpodobností převádí na pravděpodobnostní logaritmy, a proto je maximum v rovnici vypočítáno jako

$$c_{MAP} = arg \max_{c \in \{c_l, c_s\}} \left[\log P(c) + \sum_{k=1}^{m} \log P(t_k|c) \right]. \tag{3.3}$$

Naive Bayes je ve své podstatě velmi jednoduchý pravděpodobnostní model, který může být velmi snadno a také efektivně implementován s lineární složitostí.

3.2.2 LogitBoost

LogitBoost je podle [7] tzv. boostovací algoritmus, který realizuje dopředné modelování. To je založené na stupňování předchozí zkušenosti, aby se vytvořila doplňková logistická regrese. Stejně jako ostatní boostovací algoritmy, přidává iterativně základní modely a učence stejného typu. Vytvoření každého nového modelu je ovlivněno výkoností přechozích modelů. Tohoto je dosaženo tak, že se všem trénovacím vzorům přiřadí váha a během jednotlivých iterací se tato váha adaptivně upravuje. Předpokládejme, že f_m je m-tý učenec a $f_m(d)$ je předpověď hodnoty zprávy d. Poté, co je vytvořeno f_m a přidáno do souboru, jsou upraveny hodnoty trénovacích vzorků tak, že následující učenec f_{m+1} se více zaměří na vzorky, které byly pro f_m obtížné. V iteračním procesu je pravděpodobnost toho, že d je ve třídě c odhadováno aplikací logistické sigmoidní funkce, tedy

$$P(c|d) = \frac{e^{F(d)}}{1 + e^{F(d)}}, F(d) = \frac{1}{2} \sum f_m(d). \tag{3.4}$$

Jakmile je iterace skončena a je vytvořen konečný soubor F, je klasifikace cílové emailové zprávy určena pravděpodobností v rovnici (3.4).

3.2.3 Support vector machines

Tento algoritmus podle [7] používá lineární modely, aby vytvořil nelineární hranice dané kategorie tak, že provede transformací dané instance prostoru na lineárně oddělený prostor pomocí nelineárního mapování. V tomto transformovaném prostoru se vytvoří hyperrovina, která maximalizuje vzdálenost mezi trénovacími vzory dvou kategorií. Toto se provede výběrem dvou rovnoběžných hyperrovin, kde každá tato rovina je tečnou k alespoň jednomu vzorku její kategorie. Vzdálenost mezi dvěma tečnami rovin je okraj klasifikátoru, který chceme maximalizovat.

Předpokládejme, že proměnná třídy pro i-tý trénovací příklad je $c_i = \{1, -1\}$, kde 1 označuje například kategorii spamu a -1 značí druhou kategorii správných emailů. Hyperrovina v ukázkovém příkladu může být zapsána jako

$$w \cdot d + b = 0, \tag{3.5}$$

kde w je vektor, který je kolmý k hyperrovině a b je zkreslený term. Jestliže daná trénovací data jsou lineárně oddělitelná, potom můžeme vybrat dvě hyperroviny mezi kterými nejsou žádné body. Následně provedeme maximalizaci vzdálenosti mezi těmito hyperrovinami, což je $2/\|w\|$. Maximalizace okraje je ekvivalentní řešení následujícího minimalizačního problému:

$$\min_w \frac{\|w\|^2}{2}, \qquad pro\ c_i(w \cdot d_i + b) \geq 1 \tag{3.6}$$

Optimalizační problém v rovnici (3.6) se dá vyřešit Lagrangovy multiplikátory:

$$\frac{\|w\|^2}{2} - \sum_i \lambda_i [c_i(w \cdot d_i + b) - 1]. \tag{3.7}$$

Stále se ale jedná o náročný problém, protože je obsaženo velké množství parametrů. Tento problém můžeme zjednodušit transformací rovnice (3.7) na následující dvojitou formulaci, která obsahuje pouze Lagrangovy multiplikátory:

$$\max \sum_i \lambda_i - \frac{1}{2} \sum_{i,j} \lambda_i \lambda_j c_i c_j d_i \cdot d_j, \qquad pro\ \lambda_i \geq 0, a \sum_i \lambda_i c_i = 0. \tag{3.8}$$

Dvojitý optimalizační problém se obvykle řeší použitím nějaké techniky kvadratického programování, jako je sekvenční optimalizační algoritmus. Termy λ_i z rovnice (3.8) jsou použity k definování rozhodovací hranice

$$\left(\sum_i \lambda_i c_i d_i \cdot d \right) + b = 0. \tag{3.9}$$

Abychom se vypřádali s případy, kdy nemohou být trénovací vzory úplně odděleny a také s případy, kdy je povolena menší chybná klasifikace, byla vytvořena metoda na výběr měkkých hranic. Ta provádí výběr hyperrovin, které zredukují počet chyb a maximalizují šířku okraje. Je představena proměnná ξ vyjadřující stupeň špatné klasifikace:

$$\min_{w} \frac{\|w\|^2}{2} + C \sum_{i} \xi_i, \qquad pro\ c_i(w \cdot d_i + b) \geq 1 - \xi_i, \tag{3.10}$$

kde C je uživatelem specifikovaná konstanta, která určuje úroveň tolerance chybovosti.

3.2.4 Augmented latent semantic indexing spaces

Latent semantic indexing je dle [7] dobře známá technika pro získávání informací, která používá ohodnocený a redukovaný prostor. Umí efektivně transformovat jednotlivé dokumenty do sémantických vektorů, aby bylo možné provést odhad hlavních vzorů.

Rozšíření trénovacích vzorů se provádí seskupením stěžejních prvků. Pro každou kategorii emailu zkonstruujeme několik shluků. Pro každý shluk c_j se jeho stěžejní prvek vypočítá jako

$$a_{cj} = \frac{1}{k} \sum_{i=1}^{k} d_{n_i}, \quad d_{n_i} \in c_j, \tag{3.11}$$

a poté může být tento prvek použit jako reprezentace tématu v daném shluku. Jakmile jsou identifikovány stěžejní prvky kategorie c, potom se provede porovnání těchto prvků se všemi trénovacími vzory z ostatních kategorií a ty nejpodobnější se následně přidají do kategorie c.

3.2.5 Radial basis function networks

Podle [7] má typická RBF síť dopředně propojenou strukturu složenou ze tří vrstev. Jedná se o vstupní vrstvu, skrytou vrstvu nelineárního zpracování neuronů a výstupní vrstvu. Pro klasifikaci emailů má vstupní vrstva n neuronů a vstupuje do ní trénovací vzorek d. Skrytá vrstva obsahuje k výpočetních neuronů, kde každý neuron může být matematicky popsán jako RBF ϕ_i mapující vzdálenost mezi dvěma vektory v Euklidovské normě na reálné hodnoty:

$$\phi_i(x) = \phi(\|x - a_i\|_2), \qquad i = 1,2, \dots, k, \tag{3.12}$$

kde a_i jsou RBF centra ve vstupním prostoru a k je většinou menší než velikost trénovacího vzorku. Výstupní vrstva sítě má dva neurony, které produkují cílovou kategorii zprávy podle

$$c_j = \sum_{i=1}^{k} w_{ij} \phi_i(x), \qquad j = 1,2, \tag{3.13}$$

kde w_{ij} je váha spojující i-tý neuron ve skryté vrstvě s j-tým neuronem ve výstupní vrstvě. Neuronový aktivátor ϕ_i je nelineární funkce vzdálenosti. Čím je vzdálenost menší, tím je aktivátor silnější. Nejčastěji používaná bázová funkce je Gaussova

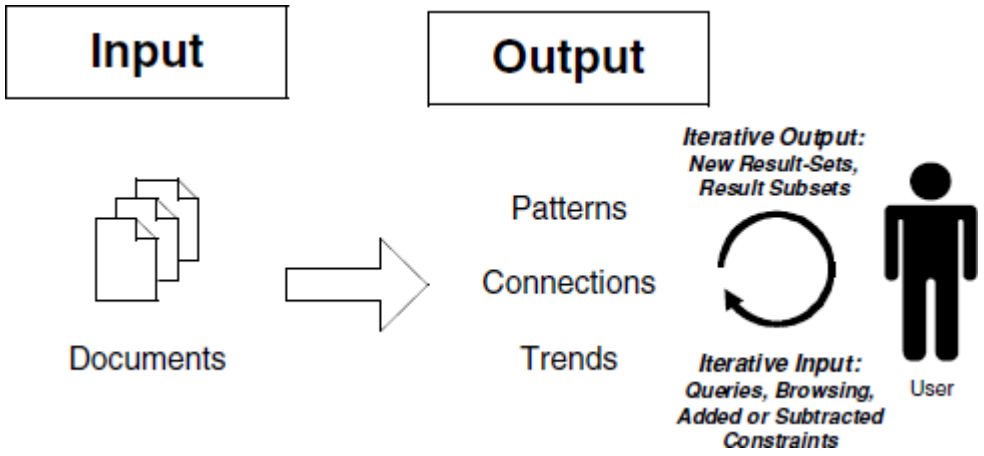
$$\phi(x) = e^{-\frac{x^2}{2\sigma^2}}, \tag{3.14}$$

kde σ je parametr šířky, který kontroluje hladkost bázové funkce.

3.3 Architektura systému dolování v textu

Důležitou částí klasifikace textu je předzpracování vstupních dat. Jak již bylo popsáno v kapitole 3.1, trénování klasifikátoru textu se provádí na zvolených dokumentech. Jestliže tedy tyto dokumenty obsahují velké množství neužitečných informací, potom může být vytvořený klasifikátor nepřesný. Stejná situace nastává i při samotném klasifikování. Přestože je vytvořený klasifikátor velmi přesný, může být problém zařadit vstupní dokument, protože obsahuje velké množství neužitečných informací. Vzhledem k těmto důvodům se nad vstupními dokumenty provádí například dolování znalostí z textu nebo další úpravy. V praxi se často používá kombinace více metod, aby se dosáhlo co nejlepších možných výsledků.

Jak uvádí [6], tak na obecné úrovni jednoduchý textový dolovací systém generuje z nezpracovaného dokumentu různé druhy výstupů, například vzory, mapy propojení nebo trendy. Tento jednoduchý princip je poté rozšířen o vstup uživatele. Obrázek 3.3, který je uvedený níže, zobrazuje tento rozšířený princip. Důležitým prvkem je právě uživatel, protože do celého procesu zasahuje a tím ovlivňuje výsledky.

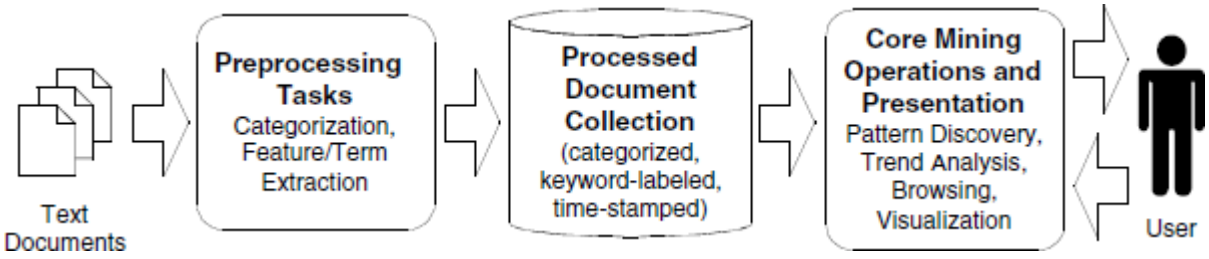


Obrázek 3.3 Iterativní model dolování v textu (převzato z [6])

Podle [6] používají systémy pro dolování textu obecného modelu, který je rozdělen na čtyři základní části. Jedná se o tyto fáze:

- **Předzpracování** – zde jsou zahrnuty všechny procedury, procesy a metody, které provádí přípravu dat ještě před tím, než se vykoná samotné dolování znalostí z textu. Tyto úkony se typicky zaměřují na předzpracování datového zdroje a kategorizaci. Obecně se tedy nejprve upraví data do kanonického formátu, odstraní se zbytečné informace a na závěr se vytvoří jednotný koncept dokumentu.
- **Dolovací operace** – jedná se o klíčovou fázi dolování v textu zahrnující objevování zajímavých vzorů, analýzu trendů a inkrementální provádění dolovacího algoritmu.
- **Prezentační vrstva** – tato část zahrnuje zobrazování výstupu uživateli. Patří sem grafické uživatelské rozhraní, možnost prohlížení dat, přístup k dotazovacímu jazyku, s pomocí kterého se provádí výběr dat, vizualizace výsledků a optimalizační nástroje
- **Zpřesňování** – na závěr se aplikují metody, které filtrují redundantní informace, více shlukují související data, ale mohou také provádět složitější operace jako je řazení, prořezávání, generalizace nebo další optimalizační postupy.

Obrázek 3.4, který je uvedený dále, zobrazuje popsané fáze obecné architektury systému pro dolování v textu. Je na něm dobře vidět, že fáze předzpracování a dolovací operace jsou nejdůležitější části celé architektury.



Obrázek 3.4 Obecná architektura systému pro dolování v textu (převzato z [6])

3.4 Extrakce klíčových slov

Podle [7] jsou klíčová slova definována jako sekvence, jednoho nebo více slov, která poskytuje stručnou reprezentaci obsahu dokumentu. V ideálním případě je celý význam daného dokumentu přesně vyjádřen za pomoci klíčových slov. Přestože poskytují nespornou výhodu při analýze, indexování a získávání dokumentů, velká část existujících dokumentů nemá definována klíčová slova.

Většina existujících přístupů extrakce klíčových slov se zaměřuje na ruční vytváření. Toto zajišťuje profesionální kurátor, který používá buď pevně danou taxonomii, nebo se spoléhá na úsudek autora. Kvalita poté záleží na autorovi dokumentu. Z těchto důvodu se výzkum soustředí na metody automatické extrakce klíčových slov. Tyto metody mohou poskytnout doporučení pro profesionálního kurátora nebo umožní vygenerovat seznam klíčových slov pro dokumenty, které je nemají.

Příkladem metody pro automatickou extrakci klíčových slov je RAKE (Rapid Automatic Keyword Extraction). Jak uvádí [7], jedná se o metodu, která nevyžaduje dohled a je doménově i jazykově nezávislá. Zjednodušeně řečeno, princip spočívá v tom, že se nejprve z dokumentu získá množina kandidátních klíčových slov. Při výběru těchto slov se provádí filtrování stop-slov na základě požadovaných parametrů. Stop-slova jsou více rozepsána v následující kapitole 3.5. Poté se z těchto slov vytvoří graf společných výskytů a pro každé kandidátní slovo se vypočítá jeho skóre. Na závěr se z uvedených slov vybere T nejlepších podle jejich skóre. Proměnná T se vypočítá jako jedna třetina všech slov uvedených v grafu.

3.5 Stop-slova

Podle [3] se jedná o množinu slov, která jsou považována za irelevantní. V anglickém jazyce se jedná například o slova jako *a, the, of, for, with* a další. V českém jazyce to mohou být slova jako *a, nebo, proto, které* atd. Důležité je, že seznam stop-slov se může lišit pro různé množiny dokumentů. Je to z toho důvodu, že například slovo databáze může být důležité v novinách, ale v dokumentech o výzkumu databáze je považováno za nežádoucí.

Seznamu stop-slov se využívá při předzpracování dokumentů, kdy jsou jednotlivá slova porovnávána oproti tomuto seznamu a když je nalezena shoda, tak je slovo vypuštěno. Zabrání se tak zbytečné indexaci neužitečných slov. Při klasifikaci textu se potom zvyšuje kvalita vytvořeného klasifikátoru, protože data nejsou zkreslena slovy bez významu.

3.6 TF-IDF

Metoda TF-IDF (anglicky Term Frequency – Inverse Document Frequency) je jedna z nejznámějších váhovacích metod, která se používá pro váhování termů. Tato technika se uplatňuje při předzpracování textových dokumentů například u klasifikace textových dokumentů.

Podle [8] určuje Term Frequency (dále jen TF) počet výskytů slov v daném dokumentu a vypočítá se pomocí následujícího vzorce:

$$TF_{i,j} = \frac{n_{i,j}}{\sum_k n_{k.j}}, \tag{3.15}$$

kde $n_{i,j}$ je počet výskytů slova T_i v dokumentu D_j a jmenovatel představuje počet slov v dokumentu D_j, tedy obecně řečeno jeho délku. Výhodou je, že TF uvažuje více frekventovaná slova v dokumentu, což je více prospěšné pro vyhledávání.

Inverse Document Frequency (dále jen IDF) určuje počet výskytů slova v rámci kolekce dokumentů s pomocí tohoto vzorce:

$$IDF_i = \log \frac{|D|}{|d_j : t \in d_j|}, \tag{3.16}$$

kde $|D|$ je celkový počet dokumentů v trénovací množině a $|d_j : t \in d_j|$ je počet dokumentů ve kterých se vyskytuje term t_i.

Potom TF-IDF spočítáme podle vzorečku:

$$TFIDF_{i,j} = TF_{i,j} * IDF_i. \tag{3.17}$$

V rámci dokumentu bude tedy vysoké TF nebo IDF, když se bude dané slovo často vyskytovat v rámci zpracovávaného dokumentu. Výsledkem bude vysoká váha.

3.7 Shrnutí

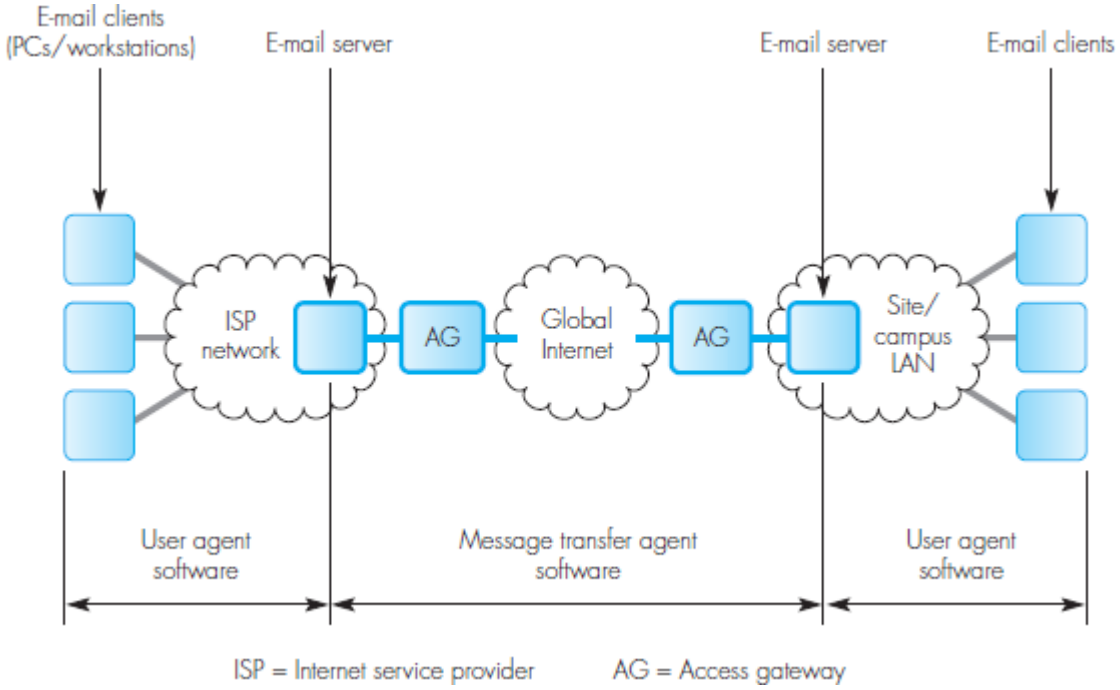
V úvodu této kapitoly je uvedeno, co je účelem klasifikace textu a jaká je souvislost mezi procesem klasifikace a získávání znalostí z databází. Dále je uvedena definice a jsou rozebrány tři základní fáze, tedy fáze trénování, testování a aplikace. Následující kapitola popisuje základní principy vybraných klasifikačních metod. Jedná se o metody Naive Bayes, LogitBoost, Support vector machines, Augmented latent semantic indexing spaces a Radial basis function networks. Poté je popsána základní architektura dolovacího systému a její souvislost s klasifikací textu. Na závěr jsou uvedeny možnosti předzpracování dokumentů. Jedná se o extrakci klíčových slov, zpracování stop-slov nebo aplikace a princip metody TF-IDF.

4 Emailová komunikace

V současné době je elektronická pošta jedna z nejpoužívanějších internetových služeb. Přestože máme velké množství způsobů jak provádět elektronicky mezilidskou komunikaci, například pomocí Facebooku, mobilního telefonu nebo nejrůznějších komunikačních programů, patří email stále k často používaným službám. Je to hlavně díky tomu, že nabízí řadu nesporných výhod, které nás motivují k jeho používání. Ať už máme firemní nebo soukromý email, je ve většině případů dostupný přes webové stránky. Není tedy nutné si instalovat žádné další doprovodné programy, ale stačí nám pouze internetový prohlížeč, který je v dnešní době již standardem na každém počítači. Poštu tak můžeme odesílat nebo číst odkudkoliv na světě. Další výhodou je, že se za odeslání nebo přijetí pošty nic neplatí a veškeré emaily se nám mohou uchovávat v naší schránce na vzdáleném serveru. Další motivací pro použití je rychlost, s jakou je email odeslán a následně přijat příjemcem. Velkou výhodou je možnost připojení příloh k odeslané zprávě. Email je proto snadným způsobem jak přenášet soubory o menších velikostech.

Záleží ovšem i na úhlu pohledu, protože hlavně u mladší generace je email nahrazován nejrůznějšími komunikačními programy, tedy icq, jabber, skype, nebo sociálními sítěmi jako je Facebook nebo Google+. Tyto prostředky nabízí nesporné výhody, jedná se například o možnost skupinové komunikace, komunikace v reálném čase nebo video hovory. I přes tyto důvody má ovšem email stále své místo a využití. Email tedy nemusí být pro mnohé lidi hlavním prostředkem elektronické komunikace, ale i tak je stále využíván.

Jak uvádí [10] je z uživatelského pohledu elektronická pošta jedna z velmi oblíbených a často používaných internetových služeb.



Obrázek 4.1 Přenos emailu přes internet (převzato z [10])

Obrázek 4.1 zobrazuje přenos emailové zprávy přes internetovou síť. Základem jsou dvě hlavní části, jedná se o emailového klienta a server. Klient je často počítač nebo pracovní stanice, na které běží

program nazývaný user agent (UA). Ten poskytuje uživatelské rozhraní pro emailový systém a umožňuje vytvořit, odeslat a přijmout emailové zprávy. Emailový server je počítačový server spravující poštu pro všechny uživatele nebo klienty, kteří jsou na něm zaregistrovaní. Je na něm také umístěný UA software pro zprostředkování komunikace s UA klientem a message transfer agent (MTA) pro odesílání a příjem zpráv na další emailové servery, které jsou přímo připojeny k internetu.

4.1 Vývoj

Období vývoje emailu je podle [9] poměrně složité, protože elektronická pošta je ve své podstatě přirozeným výsledkem vývoje síťové komunikace. Z tohoto důvodu se dá říci, že počátky vzniku emailu jsou ještě před vznikem samotného internetu. Email tak jak ho známe dnes, vznikl ale až v pozdější době.

Na počátku 60. let minulého století dochází k rozvoji počítačů se sdílením času, které umožňují běh více programů současně. Mimo jiné začínají vznikat programy na výměnu textových zpráv nebo komunikaci v reálném čase. Jednalo se tedy o přirozené využití nových možností.

Až počátkem 70. let minulého století byl odeslán první email. Koncem roku 1971 vytvořil Ray Tomlinson aplikaci schopnou přenášet zprávy přes síť ARPANET. Jednalo se o program, který měl jednoduchou funkcionalitu a byl ovládaný z příkazové řádky. Stanovil ovšem dodnes používaný model, tedy že emailová zpráva je poslaná ze zdrojové schránky odesílatele do cílové schránky příjemce. Aby se rozšířila možnost adresace v síti, použil Tomlinson symbol @ pro oddělení názvu uživatele a názvu stroje. Vznikl tak formát emailové adresy *name@host*, který je používán dodnes.

Roku 1972 byly do protokolu FTP (File Transfer Protocol) přidány příkazy MAIL a MLFL za účelem poskytnutí podpory pro přenos emailů. Další změna ohledně protokolu následovala až počátkem 80. let, kdy se přechází z FTP protokolu na efektivnější SMTP protokol (Simple Mail Transfer Protocol). Mimo jiných vylepšení poskytoval tento protokol možnost poslat jednu zprávu do domény s více než jedním adresátem a až cílový server provedl vložení zprávy do daných schránek. Více informací o SMTP protokolu je uvedeno v kapitole 4.5.1.

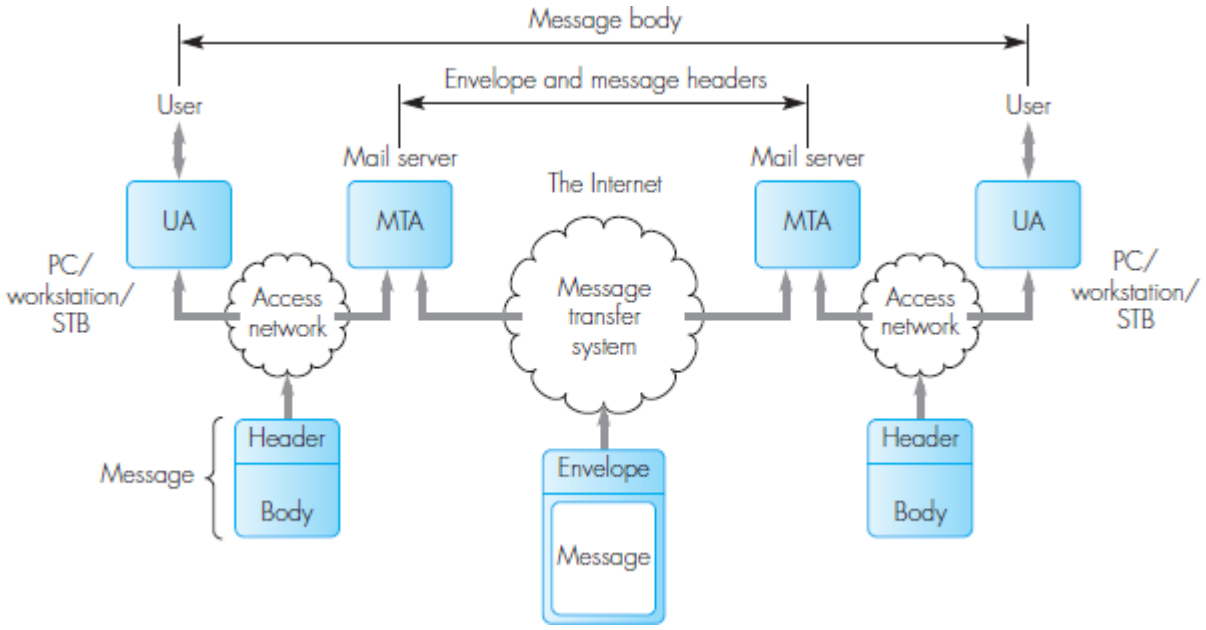
Rok 1977 přinesl sjednocení formátu emailových zpráv v podobě RFC 733, které bylo v roce 1982 nahrazeno RFC 822. Jedná se o standard popisující formát přenášených zpráv. Mimo jiné se jednalo o první standard, který popisoval syntaxi doménových jmen. RFC 822 bylo v roce 2001 nahrazeno upraveným RFC 2822 a ten byl dále v roce 2008 nahrazen dnes aktuálně platným a používaným RFC 5322.

Podle [10] bylo ke standardům popisujícím formát přenášených zpráv doplněno rozšíření, nazvané MIME (Multipurpose Internet Mail Extension). To umožňuje posílat zprávy s binárními a multimediálními daty nebo různými jazyky či specifickými znaky. Toto rozšíření je specifikováno ve standardu RFC 1341 a později bylo doplněno v RFC 2045 a RFC 2048. MIME je dále více rozebráno v kapitole 4.3.

4.2 Přenos emailu

Odeslání elektronické pošty podle [10] zahrnuje dvě důležité a oddělené procedury. V první části je potřeba vytvořit samotnou zprávu, tedy vyplnit nezbytné údaje. Jedná se například o jméno a adresu odesílatele a příjemce a také napsat posílanou zprávu. Oba tyto úkony se většinou provádí přes UA (User Agent), což je poštovní klient, který zpracovává zprávy u uživatele. Ve druhé části se provede zaobalení celé vytvořené zprávy do elektronické obálky, anglicky nazývané envelope. Tato obálka

obsahuje adresu odesílatele a příjemce a je přenesena přes síť k příjemci za pomoci MTA (Message Transfer Agent). Jedná se o server, který se stará o odesílání a doručování zpráv. Obrázek 4.2, který je uveden dále, přehledně zobrazuje celý proces přenosu emailové zprávy.



Obrázek 4.2 Proces přenosu emailu (převzato z [10])

Emailová zpráva je tedy rozdělena na dvě důležité části. Jedná se o hlavičku, která je více popsána v kapitole 4.3, a tělo emailu, popsané v kapitole 4.4. Obě tyto části jsou odděleny prázdným řádkem a obsahují specifické informace.

4.3 Hlavička zprávy

Podle [10] se hlavička emailové zprávy skládá z několika polí, která nejsou všechna povinná a také mnohá z nich jsou volitelná. Jak již bylo zmíněno v kapitole 4.2, je zadávání těchto dat většinou prováděno přes UA program. Každý výrobce tohoto programu si ovšem může zvolit volitelná pole podle svého uvážení a proto se tyto položky mohou lišit u odesílatele a příjemce. Díky RFC 5322, které specifikuje standardní formát všech položek hlavičky, by se nemělo stát, že si s daným emailem UA neporadí. Podle [11] se každé pole sestává z jednoho řádku ASCII textu. Na začátku řádku je uveden standardizovaný název pole ukončený dvojtečkou a za ní následuje hodnota pole. Hlavička tedy obsahuje tato pole, která jsou použita při přenosu zprávy:

- **From:** Emailová adresa osoby, která vytvořila zprávu.
- **To:** Emailové adresy hlavních příjemců.
- **Cc:** Seznam emailových adres dalších příjemců.
- **Received:** Cesta přes síť od odesílatele až k příjemci.
- **Return-Path:** Název posledního MTA.

Dále se v ní nachází pole používaná UA a uživatelem:

- **Sender:** Emailová adresa odesílatele zprávy.
- **Date:** Datum a čas odesláni zprávy z UA.

- **Message-Id:** Unikátní identifikátor zprávy přiřazený UA.
- **Reply-To:** Emailová adresa, na níž se má odeslat odpověď.
- **Subject:** Titulek zprávy.

Nakonec může obsahovat pole, která začínají prefixem „X-“, což jsou pole definována uživatelem, nebo programy. Příkladem může být:

- **X-PhoneNumber:** Telefonní číslo odesílatele.
- **X-FaxNumber:** Číslo faxu odesílatele.

V kapitole 4.1 bylo zmíněno rozšíření MIME, jehož hlavním účelem je poskytnout uživateli podporu při odesílání alternativních typů dat při zachování stávajícího systému přenosu. MIME tedy poskytuje další pole hlavičky, která specifikují typ dat v těle zprávy. Jedná se o:

- **MIME-Version:** Definuje použitou verzi MIME.
- **Content-Description:** Krátký popis obsahu zprávy.
- **Content-Id:** Unikátní identifikátor přirazený UA.
- **Content-Type:** Specifikuje typ dat v těle zprávy.
- **Content-Transfer-Encoding:** Použitá přenosová syntaxe.
- **Content-Length:** Počet bytů v těle zprávy.

4.4 Tělo zprávy

RFC 5322 mimo jiné definuje přesnou podobu těla emailové zprávy. Podle [11] tedy může obsahovat řádky ASCII textu s maximální délkou každého řádku 1000 znaků. Odesílající UA provede konverzi každého znaku do NVT ASCII jako přenosovou syntaxi s každým znakem převedeným do 7bitové formy. Tato procedura zajistí, že se v těle zprávy nemůže vyskytnout klíčové slovo, které by MTA považovalo za text protokolu. Jak již bylo popsáno v kapitole 4.1 a 4.3, je možné s emailovou zprávou odeslat nejrůznější multimediální obsah.

4.5 Protokoly

Tato kapitola uvádí použité protokoly pro elektronickou poštu. Jedná se o protokol SMTP, který zajišťuje přenos zpráv a protokoly POP3 a IMAP starající se o správu emailů. Na závěr je uvedeno porovnání protokolů POP3 a IMAP.

4.5.1 SMTP

Jak uvádí [12], v internetu se emailové zprávy přenáší tak, že zdrojový počítač vytvoří TCP spojení s cílovým zařízením na portu číslo 25. Na tomto portu naslouchá emailová služba, která používá protokol SMTP (Simple Mail Transfer Protocol). Jedná se o jednoduchý ASCII protokol definovaný ve standardu RFC 5321 a jeho výchozím portem je TCP port číslo 25. Služba tedy přijme příchozí spojení a uloží příchozí zprávu do příslušné schránky. Jestliže zpráva nemůže být doručena, je odesílateli odesláno vygenerované chybové hlášení obsahující mimo jiné první část nedoručené zprávy. Podle [13] se po ustanovení spojení chová odesílající stroj jako klient, který čeká až mu příjemce, tedy server, odpoví. Server začíná komunikaci odesláním zprávy s jeho identitou a informací zda je připraven přijmout zprávu. Když není server připraven, klient uvolní vytvořené spojení a svůj pokus opakuje později. Jestliže je server připraven přijmout zprávu, potom klient

odešle informaci, pro koho je daná zpráva a kdo je odesílatelem. Server zkontroluje, zda daný příjemce existuje a poté vyzve klienta, aby mu zprávu poslal. Přijetí zprávy server potvrdí a vytvořené spojení se ukončí.

4.5.2 POP3

Jak uvádí [12], na počátku bylo potřeba vytvořit protokol, který umožní UA kontaktovat MTA a také stáhnout email ze serveru poskytovatele do počítače uživatele. Proto vznikl protokol POP, nyní ve verzi 3, tedy POP3 (Post Office Protocol – version 3). Protokol je definován ve standardu RFC 1939 a jeho výchozím portem je TCP port číslo 110. Podle [14] se POP3 použije, když uživatel spustí UA. Tento program kontaktuje poskytovatele schránky a vytvoří TCP spojení na portu 110. Jakmile je spojení vytvořeno, POP3 protokol komunikuje ve třech fázích. Jedná se o:

1. **Fáze autorizace** – zde je potřeba, aby se daný uživatel přihlásil.
2. **Fáze transakce** – přenáší emaily ze serveru na klientský počítač. Přenesené zprávy jsou poté na serveru označeny ke smazání.
3. **Fáze aktualizace** – v této fázi se smažou označené emaily ze serveru.

Díky tomuto postupu je zaručeno, že se nemohou ztratit emaily například chybou přenosu. Pokud by tedy ve druhé fázi došlo ke ztrátě spojení, emaily jsou stále uloženy na serveru a díky tomu je možné je obnovit. Teprve až má klient všechny potřebné zprávy uloženy, dojde k vymazání na serveru.

4.5.3 IMAP

IMAP (Internet Message Access Protocol) je definován ve standardu RFC 3501 a jeho výchozím portem je TCP port číslo 143. Tento protokol podle [15] předpokládá, že všechny emailové zprávy uživatele zůstanou na serveru na dobu neurčitou a uživatel si pouze stáhne jejich kopie do svého počítače. IMAP poskytuje rozsáhlé možnosti pro čtení emailů nebo pouze jejich částí. To je velká výhoda, protože při pomalém nebo datově omezeném připojení je možné si přečíst pouze část rozsáhlé zprávy, která může mít velké množství příloh. Stejně tak umožňuje výběr pošty, kterou si chceme stáhnout na základě nejrůznějších vlastností. Například si můžeme stáhnout pouze ty příchozí zprávy, které jsou od námi specifikovaného uživatele. Protokol dále podporuje správu více poštovních schránek na jednom serveru. Díky tomuto je pro uživatele snadné provádět správu svých emailů.

4.5.4 Porovnání POP3 a IMAP

Pro uživatele, který ke své poště přistupuje pokaždé z jednoho stejného počítače a používá jednu emailovou schránkou, je možné použít POP3 protokol popsaný v kapitole 4.5.2. Asi správně tušíme, že tento přístup je v dnešní době reálný pouze pro malé procento uživatelů elektronické pošty. Větší část uživatelů má více jak jednu emailovou schránku, například jednu pro osobní potřebu a další pro školní nebo pracovní poštu. Ke svým schránkám se potom uživatel připojuje na nejrůznějších místech. Poštu si může číst z domácího počítače, internetové kavárny, z práce nebo ze školy. V tomto případě se vyplatí využít IMAP, protože když se provede nějaká úprava emailů, například se prohlédne nepřečtená pošta, potom je tato změna vidět i po připojení z jiného počítače. IMAP protokol poskytuje řadu dalších výhod jako je možnost přístupu do poštovní schránky více uživateli současně, práce se složkami přímo na serveru nebo schopnost zobrazit záhlaví zpráv bez nutnosti stahovat texty zpráv.

Nevýhodou IMAP protokolu je, že obsahuje velké množství nejrůznějších příkazů a proto je mnohem těžší na implementaci. Oproti tomu protokol POP3 je velmi snadný na implementaci, protože poskytuje pouze základní možnosti a příkazy pro práci s emaily. Dalším problémem může být, že IMAP není podporován všemi poskytovateli a ne každý emailový program umí zpracovat všechny jeho možnosti. Vzhledem k tomu, že POP3 protokol byl první, tak jej podporují všichni poskytovatelé a emailové programy. V dnešní době je již tento problém na ústupu, protože IMAP se stává vyžadovaným a také hodně používaným. Tabulka 4.1, uvedená níže, zobrazuje přehled a porovnání základních vlastností protokolů POP3 a IMAP.

Tabulka 4.1 Porovnání protokolů POP3 a IMAP (převzato z [12])

Vlastnost	**POP3**	**IMAP**
Definice protokolu	RFC 1939	RFC 3501
TCP port	110	143
Uložení emailu	uživatelský počítač	server
Čtení emailu	off-line	off-line / on-line
Čas pro připojení	krátce	déle
Využití zdrojů serveru	minimální	velké
Více schránek	ne	ano
Zálohu schránky provádí	uživatel	poskytovatel schránky
Dobré použití pro mobilní telefon	ne	ano
Uživatelská kontrola stahování	malá	výborná
Možnost částečného stažení	ne	ano
Jsou diskové kvóty problém	ne	časem mohou být
Jednoduché na implementaci	ano	ne
Široká podpora	ano	stále se zvětšuje

4.6 Shrnutí

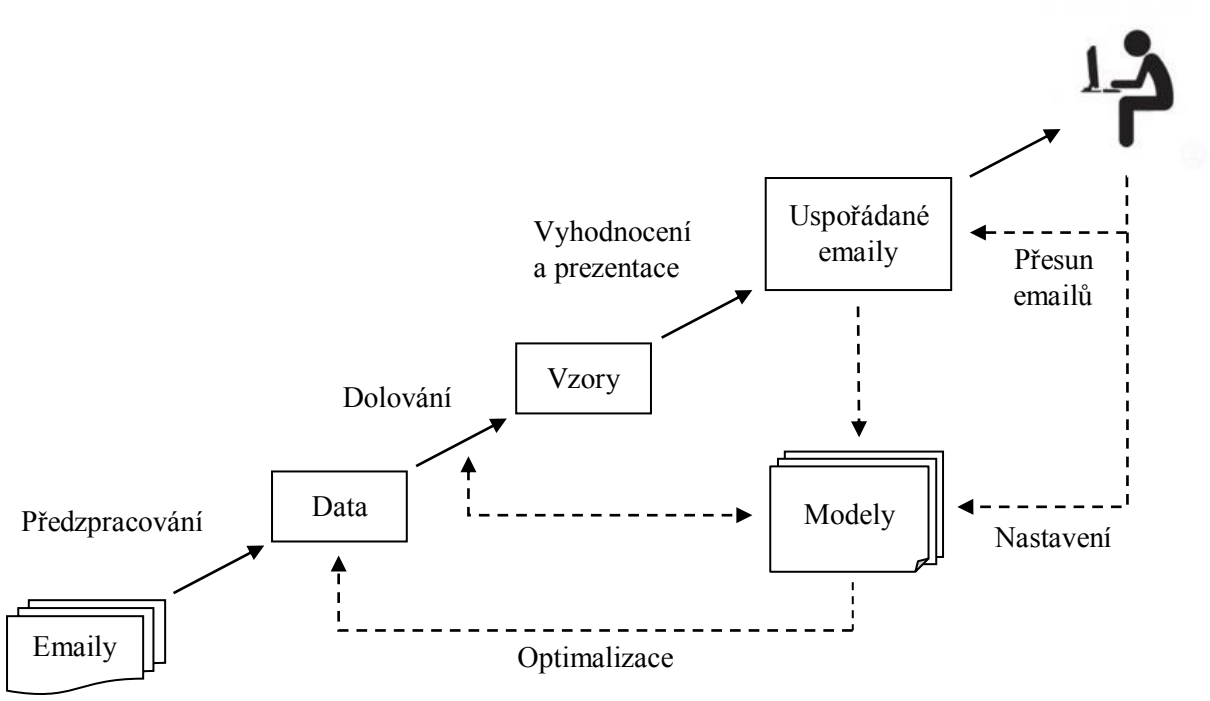
Na počátku této kapitoly je uveden základní princip přenosu emailové zprávy a na názorném obrázku jsou vysvětleny jednotlivé části přenosu. Dále je podrobně popsán historický vývoj a vznik emailové komunikace. Poté je uvedena motivace pro komunikaci za pomoci emailu a její využití. Další kapitola podrobněji rozebírá přenos emailu přes internetovou síť. Následuje rozbor samotné emailové zprávy, tedy hlavičky a těla zprávy. Na závěr jsou rozebrány příslušné emailové protokoly, což je protokol pro přenos zpráv SMTP a protokoly pro správu emailů, tedy POP3 a IMAP. Uvedené protokoly POP3 a IMAP jsou porovnány a uvedeny jejich výhody a nevýhody.

5 Návrh

Tvorba výsledného systému začíná velmi důležitou fází, kterou je návrh celého systému. Důležitost spočívá v tom, že umožňuje předcházet mnoha problémům. Ty mohou nastat, když se návrh provede jen ve zjednodušené míře nebo, v horším případě, se neprovede vůbec. Je mnohem jednodušší a hlavně rychlejší řešit problémy na úrovni návrhu než při implementaci. Poté se totiž může stát, že není možné pokračovat v implementaci nebo je nutné provést rozsáhlé změny v celém konceptu, které si vyžádají navíc zbytečné úsilí a hlavně čas, který se dá využít k vylepšení stávající funkcionality. Dobrý a kvalitní návrh je proto základem pro vytvoření dobrého a hlavně funkčního řešení. V této fázi je tedy potřeba určit co všechno bude výsledný systém provádět a dále jaké technologie budou použity.

5.1 Princip klasifikace emailů

Při tvorbě principu klasifikace emailů jsem vycházel ze základních principů klasifikace textu, které jsou uvedeny v kapitole 3. Dále jsem vycházel z principů získávání znalostí z databází, popsaných v kapitole 2. Oba tyto zmíněné procesy jsou velmi důležitým základem pro výsledný proces klasifikace množiny vstupních emailů. Obrázek 5.1, který je uvedený níže, zobrazuje celý navržený proces klasifikace emailů, jehož hlavním úkolem je roztřídit vstupní kolekci zpráv do příslušných kategorií, tedy složek uživatele. Do tohoto procesu klasifikace vstupuje na začátku kolekce emailů uživatele a na výstupu je stejná kolekce, ale uspořádaná podle klasifikačního modelu uživatele. Tento proces tedy může změnit roztřídění emailů do složek, ale také nemusí, jestliže jsou emaily správně zařazeny.



Obrázek 5.1 Návrh procesu klasifikace emailů

Proces klasifikace emailů se tedy dá rozdělit do těchto základních fází:

1. **Předzpracování** – v této počáteční fázi se provede předzpracování vstupních emailů uživatele, tedy hlavičky a těla zprávy. Hlavička je podle definice v RFC 5322 rozdělena na příslušné položky, aby se mohlo provádět oddělené vyhodnocení na základě jednotlivých částí. Takto předzpracovaná data se uloží do připravené databázové tabulky. Tímto je splněna základní myšlenka předzpracování, tedy že se data z původního zdroje převedou do vhodného formátu pro dolování. Na konci této fáze jsou emaily připraveny pro další fázi, tedy dolování.

2. **Dolování** – následuje hlavní a důležitá fáze celého procesu klasifikace, ve které se provádí samotné dolování. To ve své podstatě zahrnuje inkrementální objevování zajímavých vzorů, jejich spojování a případné provádění asociace.

3. **Vyhodnocení a prezentace** – závěrečná fáze provádí vyhodnocení a prezentaci emailů. Po této fázi jsou vstupní emaily uspořádány do příslušných složek a je vytvořen klasifikační model pro daného uživatele.

Po dokončení procesu klasifikace má uživatel možnost prohlédnout si příslušné kategorie a provést případné korekce špatně zařazených emailů nebo si vyžádat úpravu nastavení modelu. Tyto akce vyvolají úpravu vytvořeného modelu, čímž se ovlivní fáze dolování a při příští klasifikaci emailů dojde ke zpřesnění výsledků.

Při klasifikaci emailů má každý uživatel vytvořený svůj model, podle kterého se určí, do jaké složky se má daný email zařadit. Jedná se o tzv. multi-class klasifikaci, kde vstupní email patří do 0 až *n* kategorií. Pro každou složku je proto určena pravděpodobnost, s jakou do ní email patří.

V popsaném procesu klasifikace může ovšem nastat řada problémů ovlivňující výkon. Jedná se například o dlouhou dobu trénování, která může být neúnosná. Dalším problémem je přítomnost velkého počtu slov, která nemají sama o sobě význam. Tato slova způsobují šum ve vstupních datech. Při předzpracování emailů se proto použijí tzv. váhovací metody. Nejznámější metoda je TF-IDF popsaná v kapitole 3.6. Další možností bude využití seznamu stop-slov, popsaném v kapitole 3.5. Tento list obsahuje nevýznamová slova, která se při předzpracování odstraní.

5.2 Dolovací nástroj

Jak již bylo popsáno v předchozí kapitole 5.1, klíčovou fází při klasifikaci emailů je samotné dolování z textu. V rámci návrhu je proto důležité vybrat co nejvhodnější dolovací nástroj, který se bude starat o získávání znalostí z textu. Špatným výběrem tohoto nástroje by se mohly podstatně zhoršit výsledky celého procesu klasifikace. Výsledný systém by ovšem měl být uzpůsoben tak, že se tento nástroj bude dát nahradit jiným. Tato situace může nastat, protože v rámci návrhu se nedají odhalit všechny nedostatky dolovacího nástroje. Určité problémy se mohou objevit až při implementaci nebo testování a potom bude záležet na tom, jak závažné tyto nedostatky budou. Z tohoto vyplývá, že tento program nesmí být přímo integrován do systému, ale budou pouze volány jeho funkce a komponenty, které budou vracet požadované výsledky, tedy výstupy. Z pohledu návrhu se bude jednat o samostatnou komponentu. Díky tomuto principu bude možné dolovací nástroj kdykoliv nahradit a upravit pouze vstupy a výstupy na základě nově použitého programu. Na začátku je tedy velmi vhodné si nejprve stanovit základní požadavky pro výběr nástroje a teprve poté provést volbu výsledného dolovacího programu na základě těchto požadavků.

Základní požadavky na vhodný dolovací nástroj jsou následující:

- **Open-source** – musí se jednat o software s tzv. otevřeným zdrojovým kódem. Což nám zajišťuje technickou a legální dostupnost, tedy licenci. Ta umožňuje, při dodržení jistých podmínek, zdrojový kód prohlížet a upravovat.
- **Programovací jazyk Java** – vzhledem k tomu, že celý výsledný systém bude vytvořen v programovacím jazyce Java, tak je nutné, aby i implementace a možnosti použití daného dolovacího nástroje byly uzpůsobeny tomuto jazyku.
- **Dolování z textu** – jedním ze základních požadavků je široká podpora dolování z textu a dále také možnost rozsáhlé práce se vstupním textem.
- **Klasifikační metody** – samotný nástroj pro dolování by měl obsahovat a podporovat klasifikační metody jako Naive Bayes nebo Support vector machines a další.
- **Stand-alone** – velkou výhodou by byla možnost práce s nástrojem ve stand-alone módu, tedy jako samostatným programem bez nutnosti ho spouštět s pomocí podpůrných nástrojů.
- **Integrace** – dalším základním požadavkem je možnost integrace do vlastního řešení. Může se jednat například o formu knihovny, která se pouze přidá ke zdrojovým souborům, a poté se již mohou využívat jednotlivé funkce dolovacího nástroje.
- **Databáze** – nedílnou součástí musí být možnost komunikace s nějakým databázovým systémem, protože právě databáze bude jednou z klíčových komponent výsledného systému.

Po specifikaci základních požadavků je možné se blíže podívat na dostupné open-source nástroje pro dolování znalostí z textu. Příkladem mohou být tyto uvedené nástroje:

1. **Carrot2** – automaticky organizuje malé kolekce dokumentů do tematických kategorií. Jeho hlavním zaměřením jsou výsledky vyhledávání, ale umí i ostatní zdroje dat. Tento nástroj je dostupný na adrese: http://project.carrot2.org/

2. **GATE** – původně vyvinutý na University of Sheffield počátkem roku 1955. Jedná se o nástroj pro zpracování přirozeného jazyka zahrnující extrakci informací z mnoha různých jazyků. Nachází se na adrese: http://gate.ac.uk/

3. **Apache OpenNLP** – jedná se o toolkit založený na strojovém učení pro zpracování textu v přirozeném jazyce. Dostupný je z adresy: http://opennlp.apache.org/

4. **Natural Language Toolkit (NLTK)** – je soubor knihoven a programů pro symbolické a statistické zpracování přirozeného jazyka. Nachází se na adrese: http://www.nltk.org/

5. **RapidMiner** – toto prostředí zahrnuje strojové učení, dolování z dat, dolování z textu, prediktivní analýzu a obchodní analýzu. Je dostupný z adresy: http://rapid-i.com

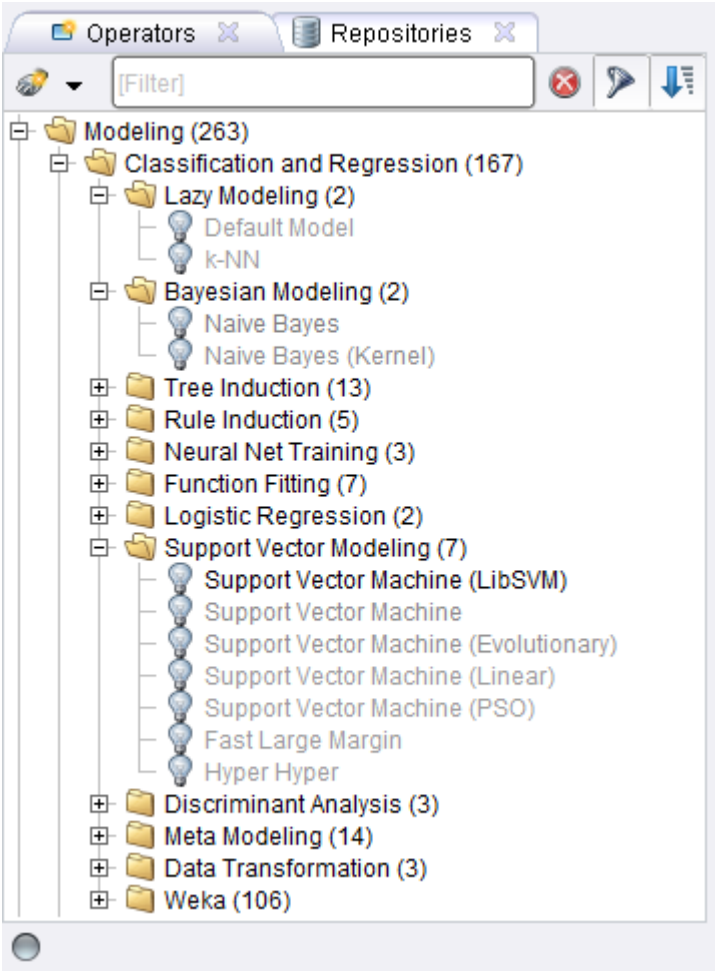
6. **Unstructured Information Management Architecture (UIMA)** – vyvinutý firmou IBM je framework k analýze nestrukturovaného obsahu jako je text, audio nebo video. Nachází se na adrese: http://uima.apache.org/

Na základě zhodnocení všech možností uvedený nástrojů jsem si zvolil program RapidMiner, který nejlépe vyhovuje všem stanoveným požadavkům. Tento nástroj bude více popsán v kapitole 6.3.

5.3 Klasifikační metody

Po výběru vhodného textového dolovacího nástroje, provedeném v předchozí kapitole 5.2, se můžeme zaměřit na klasifikační metody, které RapidMiner poskytuje. Z těchto metod se poté bude vycházet při implementaci výsledné klasifikace emailů a také při experimentech, aby se zjistilo, jaká z nabízených metod poskytuje nejlepší výsledky v závislosti na čase trénování. Obrázek 5.2, který je uvedený níže, zobrazuje ukázku vybraných klasifikačních metod programu RapidMiner. Jsou zde uvedeny tyto metody:

- **k-NN -** algoritmus k-nejbližších sousedů strojového učení pro rozpoznávání vzorů.
- **Naive Bayes -** popsaný v kapitole 3.2.1.
- **Support Vector Machine -** je rozebrán v kapitole 3.2.3.



Obrázek 5.2 Metody klasifikace RapidMineru

5.4 Emailový server

Dalším klíčovým prvkem při návrhu výsledného systému je volba vhodné implementace emailového serveru, který se bude starat o správu emailových schránek všech uživatelů. Dále bude volat komponenty zajišťující vytvoření klasifikačního modelu nebo samotnou klasifikaci s pomocí již vytvořeného modelu. Komponenty zajišťující práci s klasifikačními modely budou zahrnuty v dolovacím nástroji RapidMiner, který byl vybrán v kapitole 5.2. Zvolený emailový server proto bude základní komponenta, která bude řídit celý výsledný systém. U tohoto serveru je nutné, aby měl možnost běžet pod různými operačními systémy, protože je dobré dodržet univerzálnost celého řešení. Z uvedených věcí proto plyne, že základní požadavky na vhodný emailový server jsou následující:

- **Open-source** – musí se jednat o software s tzv. otevřeným zdrojovým kódem. Což nám zajišťuje technickou a legální dostupnost, tedy licenci. Ta umožňuje, při dodržení jistých podmínek, zdrojový kód prohlížet a upravovat.

- **Programovací jazyk Java** – vzhledem k tomu, že celý výsledný systém bude vytvořen v programovacím jazyce Java, tak je nutné, aby i implementace a možnosti použití emailového serveru byly uzpůsobeny tomuto jazyku.

- **Databáze** – nedílnou součástí musí být možnost komunikace s nějakým databázovým systémem, protože právě databáze bude jednou z klíčových komponent výsledného systému.

- **IMAP protokol** – tento protokol byl více popsán v kapitole 4.5.3 a byl také podrobně porovnán s protokolem POP3 v kapitole 4.5.2. Vzhledem k těmto poznatkům se bude komunikace s uživatelem provádět právě s pomocí protokolu IMAP. Emailový server proto musí umět tento protokol zpracovat.

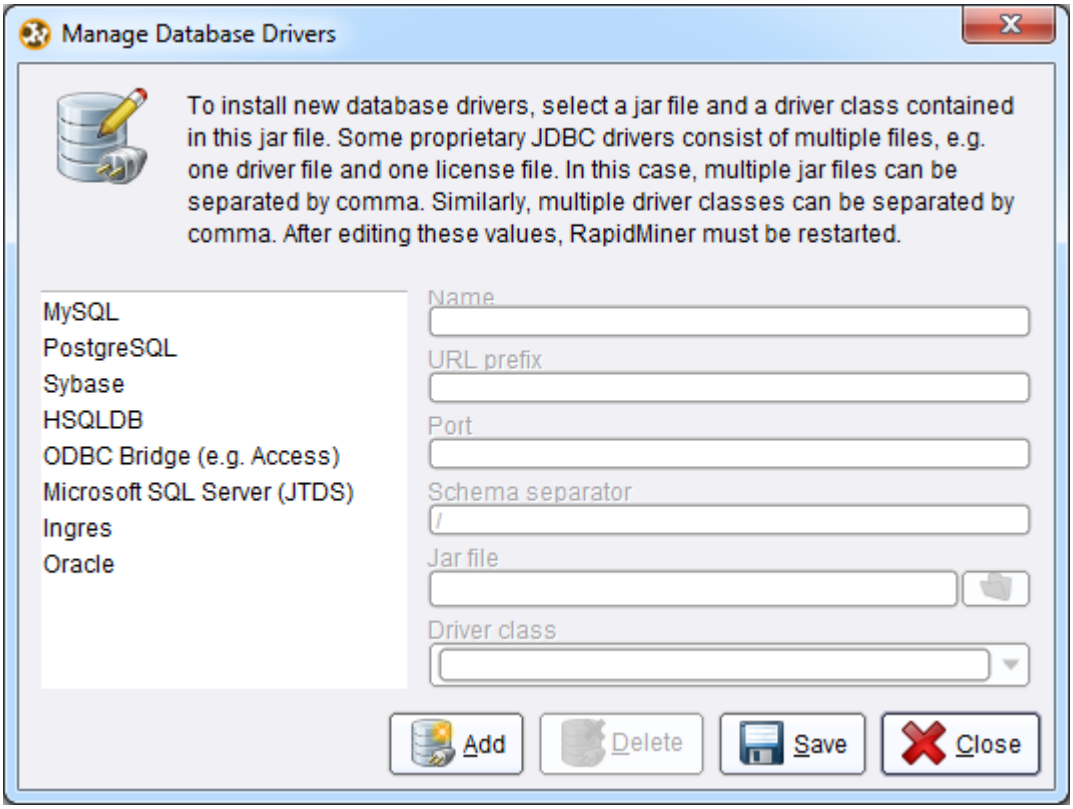
- **Podpora třídění pošty** – v ideálním případě by měl mít emailový server možnost snadného přidání vlastního modulu nebo rozšíření, které by umožňovalo pracovat s příchozími emailovými zprávami.

Na základě těchto požadavků jsem si vybral open-source implementaci emailového serveru Apache James Server, dostupného z http://james.apache.org. Jeho výhodou je, že je open-source, psaný v jazyce Java, pro ukládání pošty má široké možnosti nejrůznějších databází, podporuje IMAP protokol a má možnost implementace rozšíření pro zpracování příchozí pošty.

5.5 Databáze

Z pohledu výběru použitých technologií v rámci návrhu, je volba vhodného databázového systému posledním krokem tohoto procesu. Je to z toho důvodu, že je potřeba vzít v potaz možnosti zvoleného dolovacího nástroje – RapidMineru a vybraného emailového serveru - Apache James Serveru. Proto je nutné prozkoumat, jaké databázové systémy podporuje RapidMiner a jaké podporuje James Server. Ze shodných databázových systémů se provede výběr výsledné databáze, která se použije pro navrhovaný klasifikační systém. V databázi budou ve výsledku uložena data všech uživatelů, tedy jejich přihlašovací jména, hesla a emaily. Dále zde budou uvedeny předzpracované emaily, aby s nimi mohl pracovat RapidMiner.

Obrázek 5.3, který je uveden níže, zobrazuje databázové možnosti RapidMineru. V základu jsou podporovány databáze MySQL, PostgreSQL, Sybase, HSQLDB, ODBC Bridge, Microsoft SQL Server, Ingres a Oracle. Dále je možné doinstalovat libovolný databázový systém, stačí uvést příslušnou cestu k souboru s koncovkou *.jar* a RapidMiner si požadovaný ovladač nainstaluje. V tomto ohledu tedy dolovací nástroj nijak neomezuje výběr databáze. Jediný problém by mohl nastat s databází, která není podporována ve výchozím stavu. Jestliže by se taková databáze vybrala, musela by se pro jistotu nejprve otestovat funkčnost s RapidMinerem.



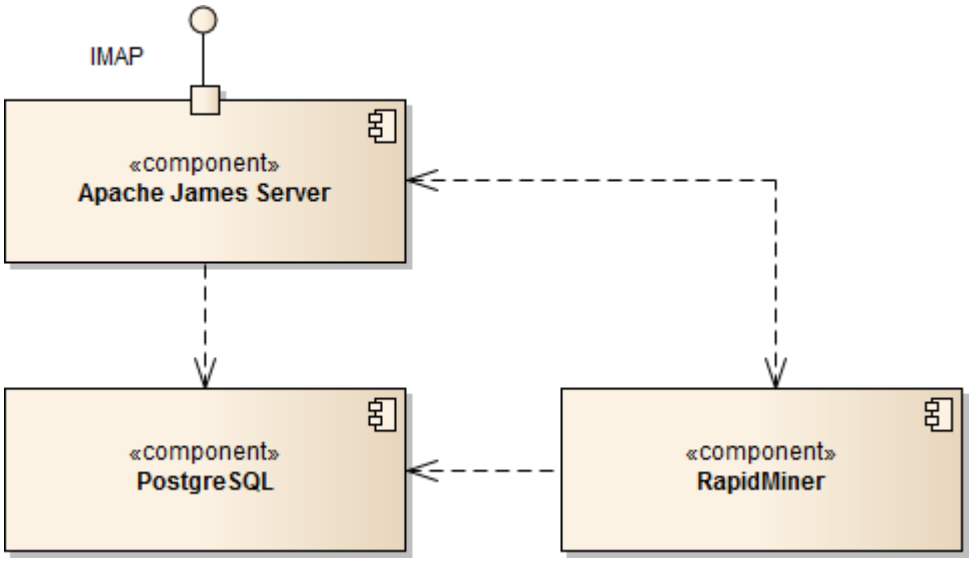
Obrázek 5.3 Databázové systémy v RapidMineru

Apache James Server rozlišuje několik základních způsobů uložení dat, podle toho kde se tato data uloží. Jedná se o soubory, kde jsou veškeré informace uloženy v souborovém systému. Tento způsob je velmi snadný na konfiguraci, ale má špatné výsledky z hlediska výkonu. Není proto doporučován pro ukládání velkého množství dat nebo pro použití v systémech závislých na výkonu. Dále mohou být data uložena v databázi, tedy ve zvoleném databázovém systému. James Server podporuje HSQLDB, MySQL, Oracle, PostgreSQL, Microsoft SQL Server a Sybase. Další možností je DBFile, což je způsob, který ukládá hlavičky zpráv do databáze a těla zpráv do lokálního souborového systému. Jeho výhodou je, že se minimalizují data uložená v databázi, ale je zachován vysoký výkon. Je také možné použít MBox, který k ukládání dat využívá speciální formát souboru. Poslední možností je ukládání v JCR (Java Content Repository).

Z nabízených možností RapidMineru a Apache James Serveru jsem si vybral databázový systém PostgreSQL. Jeho velkou výhodou je, že se jedná o open-source databázi, která poskytuje velmi dobré výkonnostní vlastnosti a má svůj jdbc driver. Jedná se o knihovnu umožňující komunikaci mezi vlastní implementací a zvoleným databázovým systémem.

5.6 Princip spolupráce

V předchozích kapitolách byly postupně vybrány jednotlivé části navrhovaného klasifikačního systému. Nyní se můžeme podívat na to, jak budou tyto komponenty mezi sebou spolupracovat a komunikovat. Obrázek 5.4, který je uvedený níže, zobrazuje diagram komponent navrhovaného systému. Základní částí je emailový server Apache James Server, se kterým se bude komunikovat s pomocí protokolu IMAP. Tento server bude dále spolupracovat s databází PostgreSQL, kde budou uložena data uživatelů, a dále s dolovacím nástrojem RapidMiner. Tento nástroj bude provádět tvorbu klasifikačního modelu uživatele a klasifikaci emailů. Částečně předzpracované emaily se budou do RapidMineru načítat z databázové komponenty.



Obrázek 5.4 Diagram komponent navrhovaného systému

5.7 Shrnutí

Na začátku této kapitoly je zdůvodněno, proč je důležité provést nejprve návrh systému a z něj poté vycházet při implementaci. Dále je navrhnut základní princip klasifikace emailů, který je zobrazen na přehledném obrázku. Následuje výběr vhodných nástrojů pro jednotlivé části celého systému. Jedná se o výběr dolovacího nástroje, kde byl zvolen program RapidMiner. Poté jsou uvedeny klasifikační metody tohoto nástroje, které se použijí při implementaci a experimentech. Dále je provedena volba vhodné implementace emailového serveru, zde byl vybrán Apache James Server. Poslední komponentou je databázový systém. Jeho volba probíhala vzhledem k možnostem RapidMineru a Apache James Serveru. Vybrána byla databáze PostgreSQL. Na závěr je uveden princip spolupráce systému za pomoci diagramu komponent.

6 Použité technologie

Po návrhu systému, který je popsaný v předchozí kapitole 5, si rozebereme základní principy použitých technologií. Tato fáze slouží k tomu, abychom se blíže seznámili s možnostmi zvolených nástrojů a technologií. V návrhu byla provedena pouze stručná analýza, která se nyní provede podrobněji. Díky tomuto kroku je možné odhalit případné nedostatky a ty vhodným způsobem vyřešit. V případě, že by byly nedostatky velmi závažné, je možné provést volbu jiného nástroje a předejít tak problémům při implementaci.

6.1 Apache James Server

Jedná se o přenosný a bezpečný emailový server, který je celý implementován v programovacím jazyce Java. Na jeho vývoji se podílí dobrovolníci z celého světa pod záštitou Apache Software Foundation. Zakladatelem projektu je Serge Knystautas.

Výhodou James Serveru je, že poskytuje modulární protokolovou architekturu a má vybudovanou infrastrukturu pro tzv. mailety. Jedná se o obdobu servletů pro webové servery, tedy doplňky umožňující jednoduše implementovat požadovanou funkčnost s využitím poskytovaného API. Právě zmíněné mailety budou hrát klíčovou roli při implementaci klasifikace emailů, protože umožňují vstoupit do celého procesu zpracování příchozí emailové zprávy. Mailety budou dále více rozebrány v následující kapitole 6.1.3.

James Server podporuje emailové protokoly SMTP, POP3, IMAP4 a LMTP. Dále umožňuje provádět SMTP autentizaci a podporuje TLS/SSL. Pro ukládání dat poskytuje podporu jak databázových systémů, tak i souborových systémů.

6.1.1 Architektura

Obrázek 6.1, který je uveden níže, zobrazuje architekturu Apache James Serveru. Základní a výchozí třídou je *server*. Tato komponenta dále komunikuje:

1. **Protokolem SMTP** – volá se třída *protocols*, která umístí email do fronty zavoláním příslušné třídy *core*. Do procesu zpracování může zasáhnout *spool*. Tato třída umožňuje vybrat email z fronty a provést zpracování nebo uložení do úložiště zavoláním třídy *store*.
2. **Protokolem IMAP** - třída *imap* provádí základní zpracování požadavků a také volá komponentu *store* pro provádění databázových operací.
3. **Protokolem POP3** – neprovádí složitější operace a proto se volá databázová třída *store*.

Dalším klíčovým prvkem je tedy *store*, zajišťující databázové operace. Tato třída podle potřeby komunikuje s následujícími komponentami:

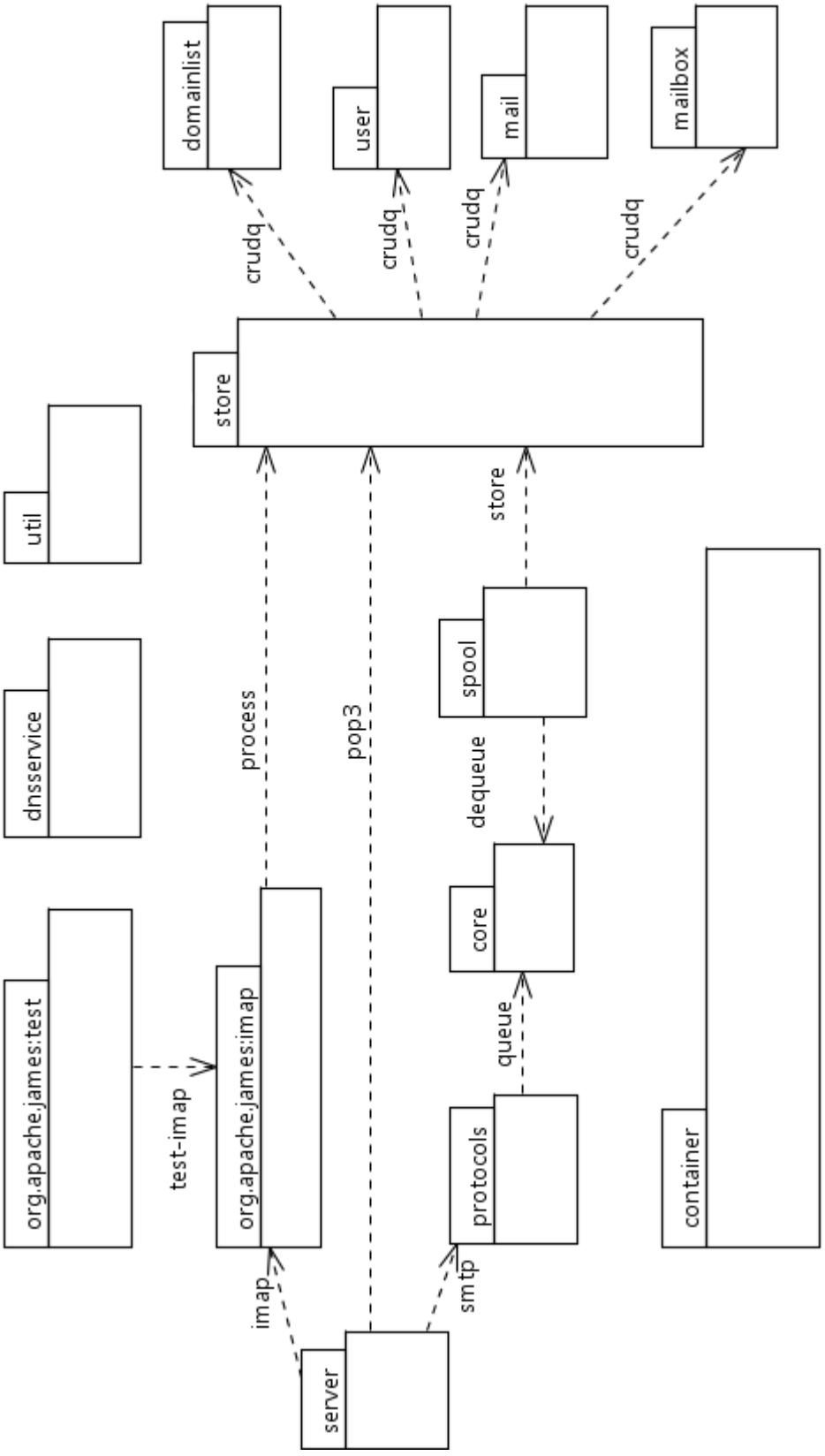
a) **Domainlist** – zpracovává spravované domény.

b) **User** – spravuje registrované uživatele.

c) **Mail** – pracuje s emaily.

d) **Mailbox** – provádí operace nad uživatelskými schránkami.

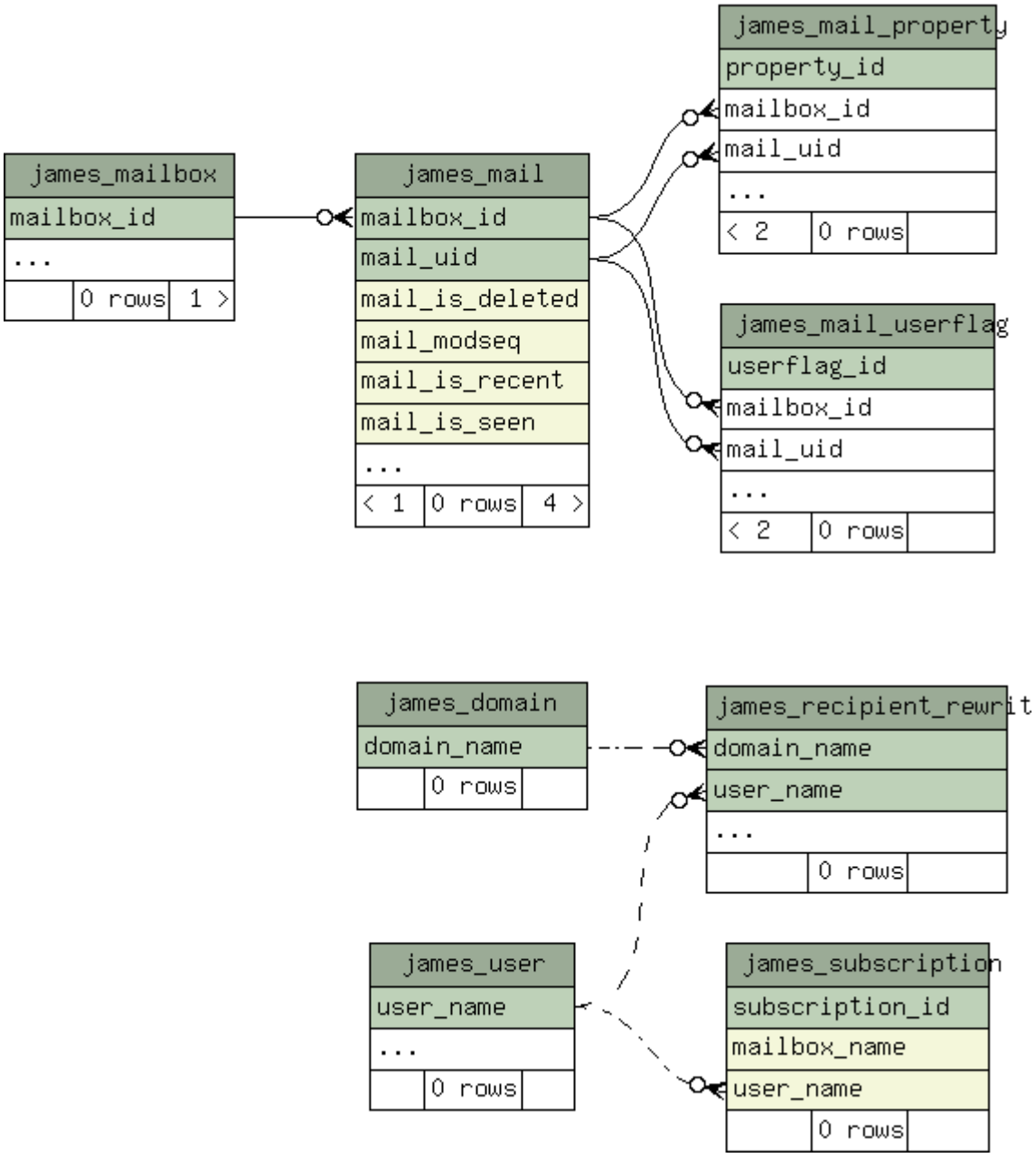
Zbývající třídy *dnsservice*, *util* a *container* poskytují potřebné utilitní metody pro ostatní části.



Obrázek 6.1 Apache James Server - architektura (převzato z [16])

6.1.2 Schéma databáze

Obrázek 6.2, který je uveden níže, zobrazuje stručné schéma databáze emailového serveru Apache James Server. Podrobnější schéma je uvedeno v příloze 1. Vidíme na něm uspořádání osmi základních tabulek v databázi a jsou zde také vyznačeny příslušné vztahy mezi jednotlivými položkami. V tabulce *james_mailbox* jsou uloženy základní informace o vytvořených složkách uživatelů. Údaje o emailových zprávách se nachází v *james_mail*. Tabulka *james_mail_property* uchovává data o vlastnostech uložených emailů. Informace o tom, jak má uživatel označeny emaily jsou uloženy v tabulce *james_mail_userflag*. Údaje o spravovaných doménách nalezneme v *james_domain*. Databázová tabulka *james_user* poskytuje data o registrovaných uživatelích. Informace o případném přesměrování emailů na jinou adresu jsou uloženy v *james_recipient_rewrite*. Tabulka *james_subscription* obsahuje údaje o tom, jaké složky má uživatel přidány v UA.



Obrázek 6.2 Apache James Server – stručné schéma databáze

6.1.3 Mailety

Jedná se o komponentu, která umožňuje pracovat s emaily. Emailový server James se snaží implementovat veškeré zpracování emailů právě do mailetů, protože se jedná o velmi flexibilní a účinný nástroj s širokým spektrem použití. Podporu při implementaci poskytuje Mailet API.

Abychom mohli vytvořit vlastní mailet, je potřeba vytvořit novou třídu, která bude dědit od abstraktní třídy `GenericMailet`. Díky tomuto principu je hlavní logika umístěna v abstraktním předkovi a my se proto můžeme soustředit na vlastní implementaci. Vytvořený mailet by měl implementovat metody `init()`, `service(Mail)`, `destroy()` a `getMailetInfo()`, ale pouze poslední metoda je vyžadována. Základní životní cyklus mailetu je následující:

1. **`init(MailetConfig)`** – vloží informace o prostředí. Zde by měly být umístěny potřebné inicializace a nastavování parametrů pro zpracování.
2. **`service(Mail)`** – jedná se o hlavní metodu, ve které se implementuje vlastní logika zpracování emailů.
3. **`destroy()`** – tato metoda se volá, když mailet zaniká.

Aby bylo možné provést logiku mailetu, je potřeba definovat kdy se má daný mailet použít. K tomuto účelu slouží tzv. matchery popsané v následující kapitole 6.1.4.

6.1.4 Matchery

Jedná se o třídu, která testuje zadanou emailovou zprávu, zda vyhovuje určitým podmínkám. Na základě výsledku se poté spustí příslušný mailet. Z tohoto tedy plyne, že pro každý mailet je definovaný určitý matcher. Samotné rozhodování nevrací informaci, zda zpráva vyhovuje nebo ne, ale vrací podmnožinu příjemců vstupní zprávy, pro které je podmínka splněna.

Existují dva způsoby implementace vlastního matcheru. První možností je vytvořit si vlastní třídu, která dědí od abstraktního předka `GenericMatcher`. Tato možnost se používá, když není potřeba vyhodnocovat adresy příjemců jednotlivě. Vlastní vyhodnocení proto vrací buď všechny příjemce, nebo nevrátí žádného. Vytvořený matcher může implementovat metody `init()`, `match(Mail)`, `destroy()` a `getMatcherInfo()`. Životní cyklus je následující:

1. **`init()`** – provádí se potřebné inicializace.
2. **`match(Mail)`** – tato metoda je povinná. Zde se provádí vlastní implementace vyhodnocení, zda zpráva vyhovuje požadovaným podmínkám.
3. **`destroy()`** – metoda se volá jestliže matcher zaniká.

Druhým způsobem je tedy vytvoření vlastní třídy, která dědí od abstraktního předka `GenericRecipientMatcher`. Zde je vyhodnocení postavené na principu testování každého příjemce zvlášť. Výsledkem je poté množina příjemců, pro které je podmínka splněna. Vytvořený matcher může implementovat metody `init()`, `matchRecipient(MailAddress)`, `destroy()` a `getMatcherInfo()`. Životní cyklus je následující:

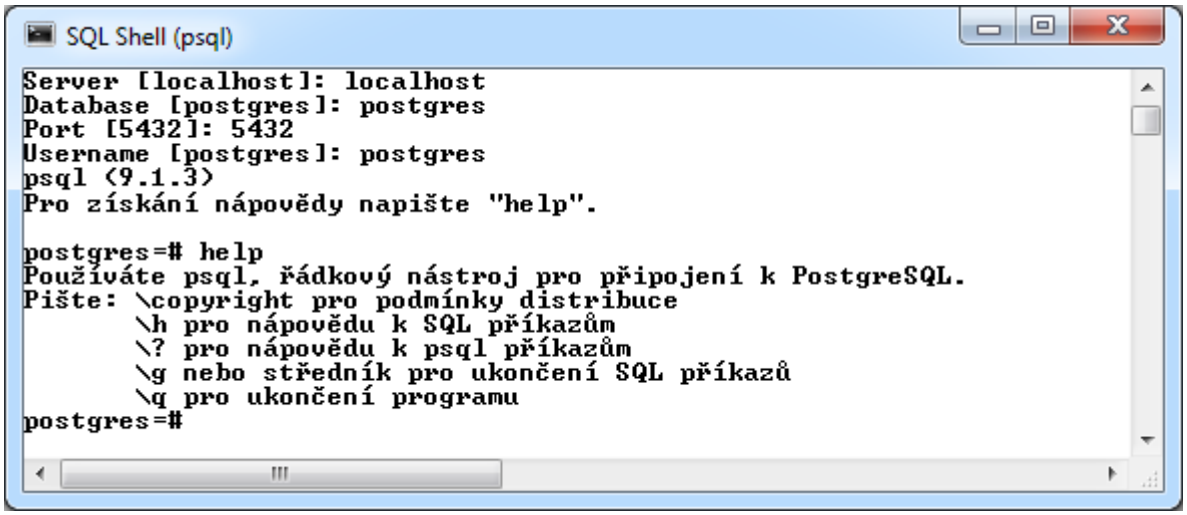
1. **`init()`** – zde je možné inicializovat potřebné prostředky.
2. **`matchRecipient(MailAddress)`** – jedná se o povinnou metodu, která určuje, zda zadaný příjemce vyhovuje stanoveným podmínkám.
3. **`destroy()`** – před zánikem matcheru je zavolána tato metoda.

6.2 PostgreSQL

Podle [17] je PostgreSQL plnohodnotným relačním databázovým systémem s otevřeným zdrojovým kódem, který má za sebou více než patnáct let aktivního vývoje. Běží nativně na všech rozšířených operačních systémech včetně Linuxu, UNIXů a Windows. Stoprocentně splňuje podmínky ACID, plně podporuje cizí klíče, operace JOIN, pohledy, spouště a uložené procedury. Obsahuje většinu SQL 92 a SQL 99 datových typů, např. integer, numeric, boolean, char, varchar, date, interval a timestamp. Vývojáři PostgreSQL se snaží o respektování a implementaci standardu ISO ANSI SQL. K systému existuje kvalitní a volně dostupná dokumentace. Výkonnostně nezaostává za srovnatelnými komerčními systémy a častokrát je i předčí. PostgreSQL je šířen pod BSD licencí, která je nejliberálnější ze všech open source licencí. Tato licence umožňuje neomezené bezplatné používání, modifikaci a distribuci PostgreSQL a to ať pro komerční nebo nekomerční využití. PostgreSQL se může šířit se zdrojovými kódy nebo bez nich, zdarma nebo komerčně.

6.2.1 psql

Primárním nástrojem pro správu databáze PostgreSQL je psql určený pro příkazovou řádku, který může být použit pro přímé zadávání SQL dotazů, nebo jejich spouštění ze souboru. Kromě toho psql poskytuje meta-příkazy a nejrůznější vlastnosti, které připomínají shell, pro zjednodušení psaní skriptů a automatizaci nejrůznějších činností. Jedná se například o automatické doplňování jmen objektů a SQL syntaxe. Obrázek 6.3, který je uvedený níže, zobrazuje ukázku popsaného psql nástroje.



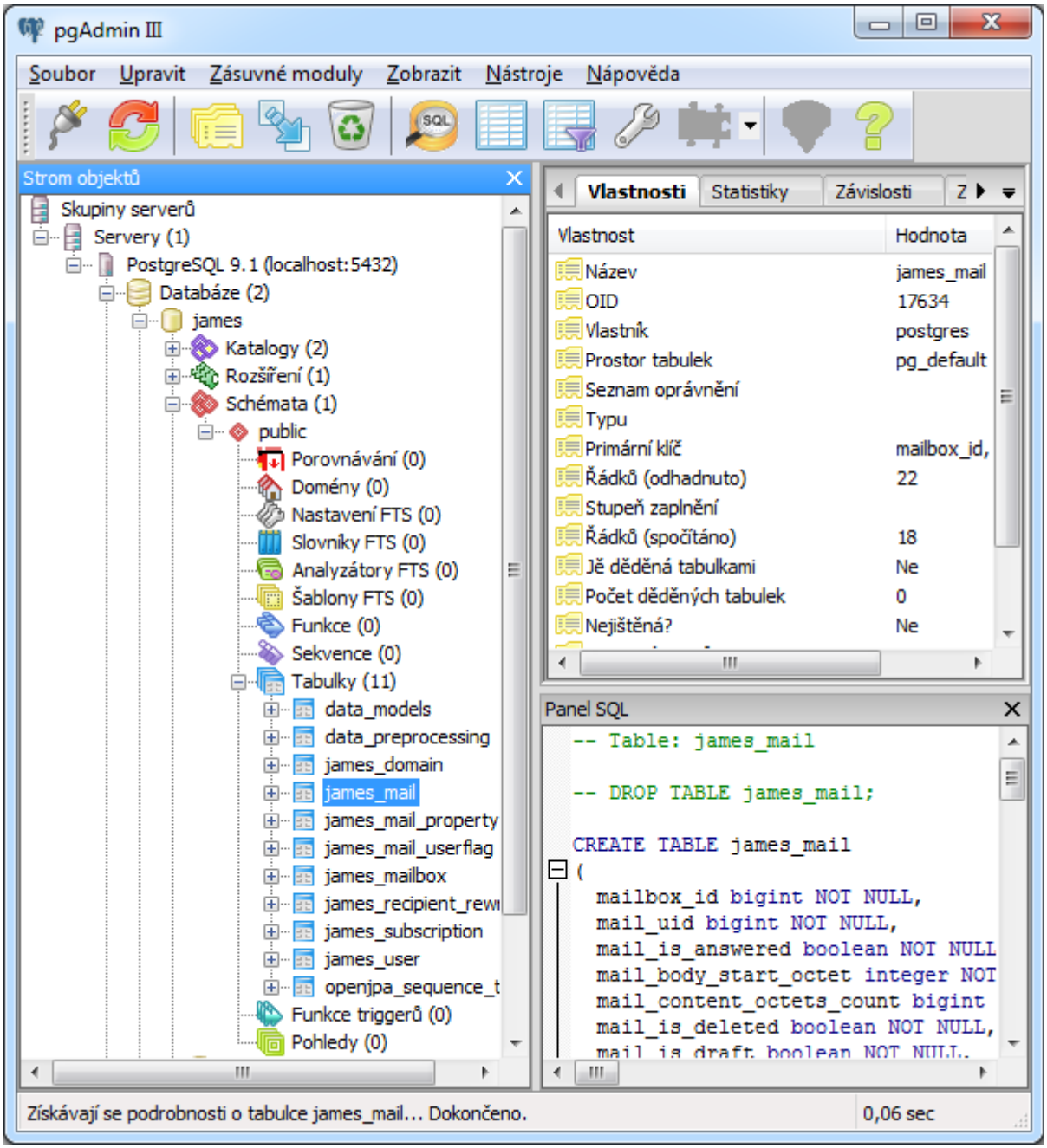
Obrázek 6.3 Nástroj psql

6.2.2 phpPgAdmin

Tato webová aplikace je napsaná ve skriptovacím jazyce PHP a slouží ke správě databázového serveru PostgreSQL. Jedná se o nástroj založený na velmi populární aplikaci phpMyAdmin, která slouží pro administraci databázového serveru MySQL. phpPgAdmin vznikl v roce 2002 jako vývojová větev phpMyAdminu s cílem poskytnout podobné funkce, ale pro databázi PostgreSQL. V současné době se jedná o dva rozdílné produkty.

6.2.3 pgAdmin

Jedná se o open-source grafické administrační rozhraní podporované na většině populárních platforem, které je přeloženo do více než dvanácti jazyků. Nejnovější je verze pgAdmin III napsaná v programovacím jazyce C++ pomocí frameworku wxWidgets, což umožňuje multiplatformní použití. Tento nástroj umožňuje jednoduše vytvářet a spravovat databáze, uživatele, tabulky, trigery a další funkce databáze PostgreSQL. Obrázek 6.4, který je uvedený níže, zobrazuje ukázku popsaného nástroje pgAdmin III na databázi Apache James Serveru.



Obrázek 6.4 Nástroj pgAdmin III

6.3 RapidMiner

Jak uvádí [18] jedná se o open-source systém pro dolování z dat implementovaný v programovacím jazyku Java. Nástroj je dostupný jako samostatná aplikace pro datovou analýzu i jako knihovna, kterou je možné použít ve vlastní implementaci. Na stránkách www.rapid-i.com je tento nástroj dostupný pod Community nebo Enterprise edicí. Community edice je dostupná zdarma pro výzkum, testování a osobní použití.

Podle [19] definuje RapidMiner každou úlohu jako proces, který se sám o sobě skládá z nejrůznějších operátorů. Tyto operátory je možné spojovat a kombinovat ve složitější struktury, se kterými lze poté provádět například klasifikaci textu nebo dolování z dat. Vytváření a testování procesu probíhá v grafickém uživatelském rozhraní. Výsledný proces je uložený v xml souboru, s pomocí kterého je možné proces spustit z vlastní implementace pomocí příslušné knihovny.

RapidMinery obsahuje skoro 700 operátorů pro nejrůznější úlohy. Tyto operátory jsou děleny do následujících kategorií:

- **Process Control** – řídí provádění procesů, například smyčky, větve nebo kolekce.
- **Utility** – pomocné operátory, jedná se například o logování, makra a jiné.
- **Repository Access** – umožňují základní práci s vnitřní repository.
- **Import** – provádí import dat.
- **Export** – provádí export dat.
- **Data Tranformation** – operátory pro transformaci dat na požadované typy.
- **Modeling** – obsahuje operátory implementující dolovací algoritmy.
- **Evaluation** – umožňují vyhodnocení nebo vizualizaci výsledků.
- **Text Processsing** – poskytuje operátory pro zpracování textových dat.
- **Web Mining** – umožňují pracovat s webovými službami nebo html stránkami.
- **Reporting** – umožňuje generovat výsledky ve formátech pdf, excel, html nebo rtf.

6.4 Shrnutí

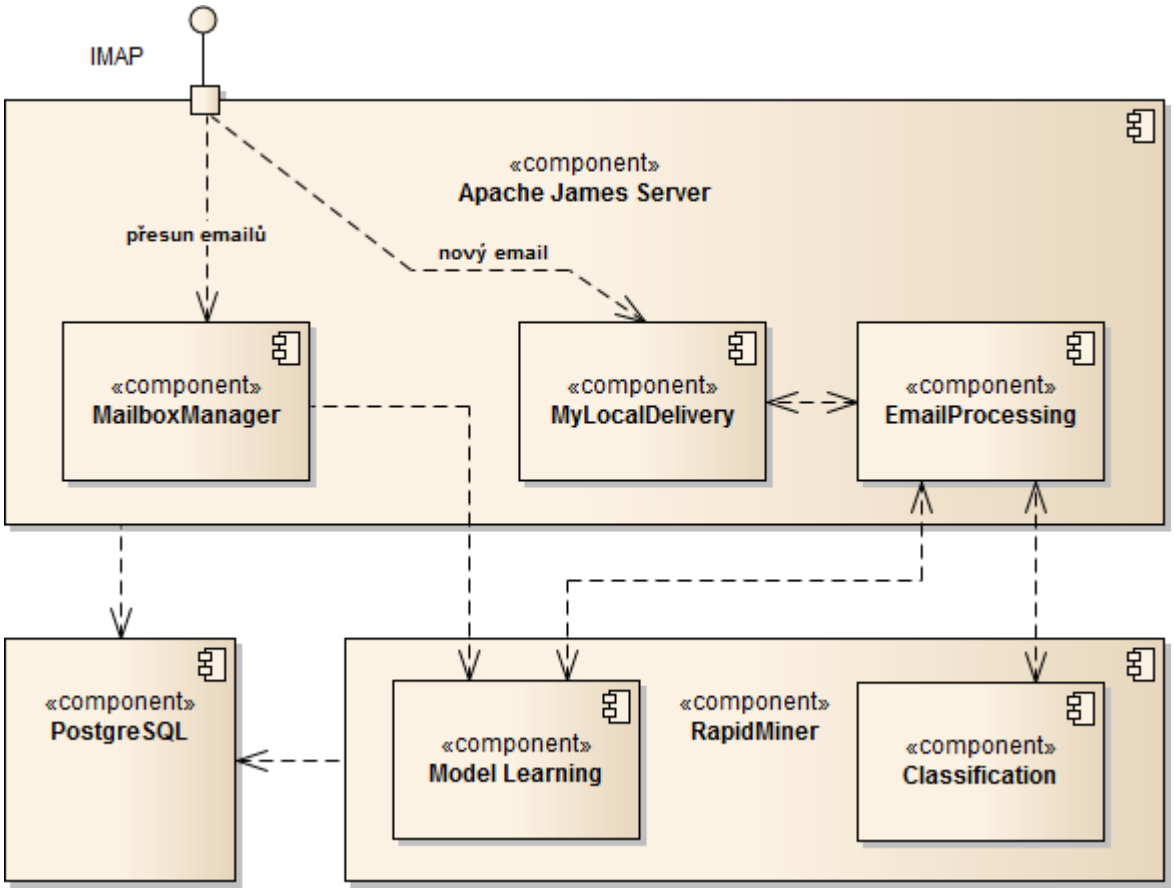
Na začátku této kapitoly bylo uvedeno, proč je důležité provádět analýzu použitých technologií. Dále byl popsán emailový server Apache James Server z pohledu architektury, databáze a možností implementace rozšíření pomocí mailetů a matcherů. Poté byl popsán databázový systém PostgreSQL s rozborem nástrojů, které se používají pro správu databáze. V závěru je uveden popis dolovacího nástroje RapidMiner.

7 Implementace

Tato kapitola se zabývá provedenou implementací emailového serveru, který provádí klasifikaci příchozí pošty. Jsou zde proto rozebrány všechny vytvořené nebo upravené části v použitých technologiích. Při implementaci se vychází z návrhu, uvedeném v kapitole 5 a také z analýzy použitých nástrojů, která je podrobně popsána v kapitole 6.

7.1 Diagram komponent

Obrázek 7.1, který je uveden níže, zobrazuje diagram komponent implementovaného klasifikačního systému. Základním prvkem je emailový server Apache James Server, ve kterém jsou provedeny potřebné a nezbytné úpravy. Jedná se především o rozšíření existující třídy *MailboxManager*, která má na starosti operace se schránkou uživatele. Do této třídy se přidala nová metoda volaná při přesunu emailů, která uloží do databáze informaci, že se změnil klasifikační model uživatele. Více je tento princip popsán a zdůvodněn v následující kapitole 7.2.3. Na obrázku jsou dále uvedeny komponenty *MyLocalDelivery* a *EmailProcessing*. Jedná se o dvě nově implementované části, které mají za úkol provést vyhodnocení a klasifikaci emailů při příchozím emailu. Následující kapitoly 7.2.1 a 7.2.2 popisují podrobněji jejich implementaci a podmínky vyhodnocení.



Obrázek 7.1 Diagram komponent implementovaného systému

Druhou hlavní částí je dolovací nástroj RapidMiner. Ten obsahuje vytvořenou komponentu *ModelLearning*, která má na starosti vytváření klasifikačního modelu pro daného uživatele. Dále je zde nová komponenta *Classification* pro samotnou klasifikaci emailů. Podrobnější rozbor tvorby modelu a prováděným procesem klasifikace se zabývá dále uvedená kapitola 7.4.

Poslední důležitou komponentou je databáze PostgreSQL, která zprostředkovává požadavky týkající se databáze. Jedná se o společný prvek, protože jak v Apache James Serveru, tak i v RapidMineru je vyžadována komunikace s databází, která probíhá za pomoci příslušného driveru. Databáze je více rozebrána v dále uvedené kapitole 7.3.

7.2 Apache James Server

V rámci implementace je potřeba provést nezbytné úpravy tak, aby emailový server Apache James Server umožňoval a prováděl klasifikaci pošty uživatele. Kapitola se proto zabývá všemi provedenými úpravami zdrojového kódu serveru.

7.2.1 Mailety

Při implementaci vlastních mailetů jsem vycházel z jejich základních principů, které jsou popsány v kapitole 6.1.3. Pro každý mailet je potřeba přidat do konfiguračního souboru *mailetcontainer.conf* informaci o tom, za jakých podmínek se má použít. Tento soubor je umístěn zde:

```
james-server/mailetcontainer-api/src/main/resources
```

Jako první jsem přidal nový mailet *MyLocalDelivery* pro doručení příchozích zpráv. Při jeho implementaci jsem vycházel z již existujícího mailetu *LocalDelivery*, který provádí pouze jednoduché doručení zprávy do složky pro příchozí zprávy. Zde bylo potřeba přidat vyhodnocení ohledně tvorby klasifikačního modelu a také vlastní klasifikace emailů uživatele. Do konfiguračního souboru *mailetcontainer.conf* se proto přidal následující řádek:

```
<mailet match="RecipientIsLocal" class="MyLocalDelivery"/>
```

Tento řádek vyjadřuje, že se na příchozí zprávu má použít matcher *RecipientIsLocal* a v případě pravdivého vyhodnocení se zavolá mailet *MyLocalDelivery*, aby provedl zpracování příchozího emailu. *RecipientIsLocal* vrací množinu těch příjemců zprávy, kteří jsou spravování Apache James Serverem. Obecně se tedy dá říci, že *MyLocalDelivery* se vykoná pro každého „lokálního“ příjemce emailové zprávy.

Životní cyklus a posloupnost provádění jednotlivých metod implementovaného mailetu *MyLocalDelivery* je následující:

1. **`init()`**
   a) Nejprve se inicializuje nastavení třídy `RecipientRewriteTable()`, která se stará o nastavené přesměrování emailů na jinou adresu.
   b) Poté je potřeba inicializovat třídu `SieveMailet()`. Tato třída provádí fyzické doručení emailu do lokální schránky uživatele.

2. `service(Mail)`

a) Pro každého příjemce emailové zprávy se nejprve provede vyhodnocení, zda je vytvořený klasifikační model zastaralý nebo neexistuje. Tato informace se načte z databázové tabulky *data_models*. Když je vyhodnocení kladné, je potřeba vytvořit nový klasifikační model a proto se přechází na bod b), jinak se pokračuje bodem c). Nový klasifikační model se vytváří tehdy, jestliže pro daného uživatele doposud neexistuje, nebo když uživatel prováděl přesun svých emailových zpráv.

b) `preprocessEmails(userName)` – tato metoda provádí základní předzpracování emailů na straně serveru. Akce je prováděna pouze tehdy, jestliže je potřeba pro uživatele vytvořit nový klasifikační model a proto se do předzpracování dat nezahrnuje nově příchozí email, protože ještě neznáme jeho správné zařazení. Postup předzpracování dat je následující:

   i. Z databáze se načtou jen ty emaily uživatele, které ještě nebyly předzpracovány.

   ii. Nad načtenou kolekcí zpráv se provádí cyklus, kde se každý email rozdělí na hlavičku a tělo. Hlavička zprávy je ještě dále rozdělena na jednotlivé položky.

   iii. Výsledná kolekce, takto předzpracovaných emailů, je uložena do databázové tabulky *data_preprocessing*.

c) `sieveMailet.service(mail)` – nyní se zavolá mailet pro doručení emailu do příchozí složky uživatele. Email se tedy vždy nejprve doručí a až poté je spuštěn proces klasifikace všech zpráv. Momentálně tedy není možné nejprve určit pro email vhodnou složku a do ní email přímo zařadit. Je to z toho důvodu, že server obsahuje chybu, která způsobí vícenásobné doručení zprávy při jakékoliv Java vyjímce. Ta může nastat při vytváření klasifikačního modelu nebo při samotném procesu klasifikace. Chyba je evidována v projektu James Serveru, ale bohužel ještě nebyla vyřešena.

d) `emailProcessing(recipients)` – jedná se o hlavní metodu, která se stará o řízení procesu tvorby klasifikačního modelu a také samotné klasifikace. Pro každého příjemce dané emailové zprávy se provede:

   i. `initializeRapidMiner()` – metoda provede nezbytnou inicializaci potřebných knihoven a nástrojů pro dolovací a klasifikační program RapidMiner.

   ii. `modelLearning(userName)` – nejprve se z databázové tabulky *data_models* zjistí, zda je potřeba vytvořit nový klasifikační model. Jestliže ano, potom se spustí učící proces v RapidMineru, jinak se přechází na bod iii. Princip učícího procesu je podrobně rozepsán v kapitole 7.4. Vytvořený model je uložen do složky *./models/user_name/*.

   iii. `preprocessEmails(userName)` – provede předzpracování emailových zpráv stejným způsobem, jak bylo popsáno v předcházejícím bodu b)

   iv. `classifyEmails(userName)` – s pomocí RapidMineru se načte a aplikuje klasifikační model na všechny emaily uživatele. Model nám vrací pro každý email a složku čtyři hodnoty pravděpodobnosti zařazení emailu do dané složky uživatele. Tyto hodnoty jsou následně sečteny a z výsledů je vybrána složka s největší pravděpodobností zařazení, do které je email přesunut.

Tímto je tedy popsán celý proces, který se provádí při nově příchozí emailové zprávě.

Druhý mailet, který jsem implementoval, je *ImportMailet*. Jak již název napovídá, provádí import testovací sady emailů Enron Dataset. Více je tato množina testovacích zpráv rozepsána v kapitole 8, věnované experimentům a testování klasifikace emailů. Do konfiguračního souboru *mailetcontainer.conf* se proto přidal následující řádek:

```
<mailet match="ImportMatcher" class="ImportMailet"/>
```

Tento řádek vyjadřuje, že se na příchozí zprávu má použít matcher *ImportMatcher* a v případě pravdivého vyhodnocení se zavolá mailet *ImportMailet*, aby provedl zpracování dat. *ImportMatcher* vyhodnocuje, zda mezi příjemci zprávy je emailová adresa *import@example.com*. Z tohoto proto plyne, že se import testovací sady emailů provede pouze tehdy, když se na adresu *import@example.com* pošle email v určitém tvaru. Na tělu zprávy nezáleží a může tedy být ponecháno prázdné. Důležitou položkou je předmět zprávy, do kterého se uvede uživatelské jméno uživatele ve formátu *uzivatel@domena*, pro kterého požadujeme provést import emailových zpráv. Zde je možnost zadat pouze jednoho uživatele, protože samotný import se provádí pro tisíce emailů a může trvat dlouho. Jestliže bychom chtěli provádět import více uživatelů, je možnost poslat několik emailů současně a import se bude provádět paralelně. Dokončení importu emailů je signalizováno doručením zprávy do příchozí pošty uživatele *import@example.com*.

Životní cyklus a posloupnost provádění jednotlivých metod implementovaného mailetu *ImportMailet* je následující:

`service(Mail)`

a) `getUserNameFromEmail(mail)` – jak již bylo popsáno dříve, budou se importovat emaily uživatele, který byl specifikován v odeslaném emailu. Metoda proto z příchozí zprávy získá uživatelské jméno. To je uloženo v předmětu přijaté zprávy ve formátu *uzivatel@domena*. Jestliže není uživatelské jméno zadáno, potom je provádění celého importu ukončeno.

b) Nyní se postupně prochází složky uživatelů, které obsahují jejich emaily. Když se najde shoda se zadaným uživatelským jménem, potom se pokračuje bodem i, jinak se končí.

   i. `initUser(userName)` – nejprve je potřeba zjistit, zda v databázi již existuje doména importovaného uživatele. V případě že neexistuje, je do databáze vložena jako nová. Dále se otestuje, zda existuje importovaný uživatel a v případě že ne, je založen nový s uživatelským jménem *uzivatel@domena* a heslem *uzivatel*.

   ii. `createUserSession(userName)` – při práci se schránkou uživatele se musí nejprve vytvořit uživatelská *session*, která se později použije při importu zpráv.

   iii. `importEmailsFromFiles(userName, mailbox, folder, session)` – postupně se prochází všechny souborové složky uživatele a když se narazí na soubor obsahující email, je proveden import tohoto emailu do databáze. Zprávy se vkládají do stejných složek, jako jsou uloženy na disku.

   iv. `session.close()` – na závěr je potřeba dříve vytvořenou uživatelskou *session* uzavřít a uvolnit tak přidělené prostředky.

Tímto je tedy popsán celý proces, který se provádí při importu emailů z datové sady Enron Dataset, popsané v kapitole 8.1.

7.2.2 Matchery

Při implementaci vlastních matcherů jsem vycházel z jejich základních principů, které jsou popsány a rozebrány v dříve uvedené kapitole 6.1.4.

RecipientIsLocal je matcher, který již Apache James Server obsahuje a pro potřeby mnou implementovaného mailetu *MyLocalDelivery* nebylo potřeba provádět žádné úpravy. Proto jen stručně uvedu jeho princip fungování. Tento matcher tedy vrací takovou množinu příjemců zprávy, pro kterou platí, že každý email v této množině je lokálním emailem. Podmínka je proto splněna pouze pro ty uživatele, kteří jsou registrováni v Apache James Serveru.

ImportMatcher je nově implementovaný matcher, který se používá při provádění importu emailů z datové sady Enron Dataset. Jedná se o třídu, která dědí od abstraktního předka *GenericRecipientMatcher* a tím pádem provádí kontrolu každého adresáta jednotlivě a vrací pouze ty adresy, pro které se má provést přísulšný mailet definovaný v konfiguračním souboru *mailetcontainer.conf*. Matcher obsahuje tuto hlavní metodu:

`matchRecipient(MailAddress recipient)` – provádí vyhodnocení, zda emailová adresa z parametru odpovídá emailové adrese *import@example.com*. Při shodě se vrací návratová hodnota `true`, jinak se vrací `false`.

7.2.3 Přesun emailů

Další nezbytnou úpravou Apache James Serveru je zásah do procesu, který provádí přesun emailů na základě požadavků od uživatele. Touto činností se zabývá rozhraní *MailboxManager*. Konkrétní implementace pro IMAP protokol je ve třídě *StoreMailboxManager*. Do této třídy jsem proto přidal novou metodu:

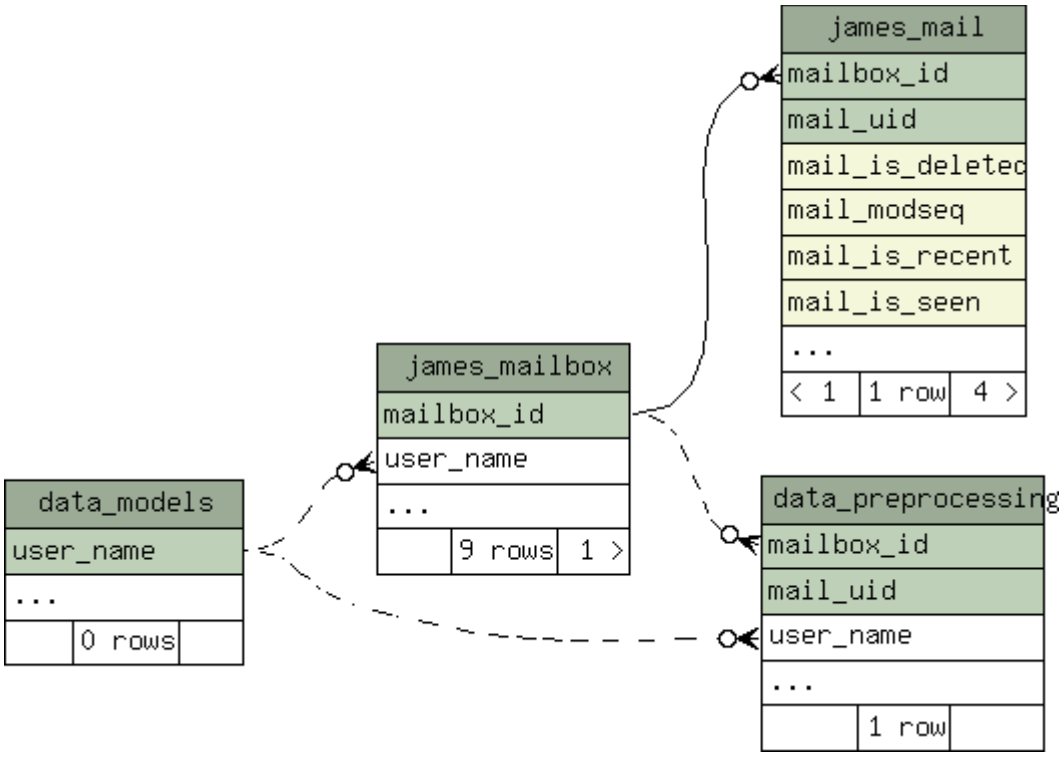
`changeUserModel(userName)` – vstupním parametrem je jméno uživatele, pro kterého se má zneplatnit vytvořený klasifikační model. Metoda tedy zavolá databázovou vrstvu a ta vloží do tabulky *data_models* informaci o tom, že u zadaného uživatele je aktuální model zneplatněn.

Samotný přesun emailových zpráv pomocí IMAP protokolu provádí třída *CopyProcessor*. Ta obsahuje metodu `doProcess(…)`, která implementuje přesun pošty ze zdrojové do cílové složky uživatele. Zde se tedy získá instance *MailboxManageru*, konkrétně *StoreMailboxManageru* a zavolá se nově vytvořená metoda `changeUserModel(userName)`. Touto akcí se zneplatní klasifikační model a při nově příchozím emailu se vytvoří nový. Díky tomuto postupu je zaručeno, že se nový klasifikační model nebude vytvářet okamžitě, jak uživatel začne přesouvat emaily, ale až v době kdy je potřeba nový model uplatnit.

7.3 Databáze

Základní struktura databáze emailového serveru Apache James Server je popsána v kapitole 6.1.2. Je zde názorně zobrazeno schéma databáze. Toto schéma bylo nutné rozšířit o další dvě nové tabulky, které poskytují podporu při klasifikaci emailů a tvorbě klasifikačního modelu. Obrázek 7.2, který je uveden níže, zobrazuje zjednodušenou část schématu databáze, kam byly přidány nové tabulky *data_models* a *data_preprocessing*. Jsou zde také znázorněny nově vytvořené vazby na původní tabulku *james_mailbox*. Podrobné schéma je uvedeno v příloze 1.

- **data_models** – jedná se o jednoduchou tabulku, která uchovává informaci o tom, zda je potřeba obnovit klasifikační model uživatele. Obsahuje proto tyto dva sloupce:
    1. `user_name character varying(200) NOT NULL` – jedná se o primární klíč obsahující uživatelské jméno.
    2. `refresh boolean` – obsahuje hodnotu `TRUE` jestliže se má změnit klasifikační model nebo `FALSE` když model vyhovuje.
- **data_preprocessing** – tato databázová tabulka byla přidána, aby uchovávala předzpracované emaily uživatelů. Obsahuje proto tyto sloupce:
    1. `mailbox_id bigint NOT NULL` – jedná se o primární klíč obsahující id uživatelské schránky. Pomocí tohoto id můžeme odlišit jednotlivé složky uživatele.
    2. `mailbox_name character varying(200) NOT NULL` – primární klíč, který určuje, jak se jmenuje příslušná složka uživatele.
    3. `mail_uid bigint NOT NULL` – unikátní id emailové zprávy. Díky tomuto id můžeme přesně určit, o jaký email se jedná.
    4. `user_name character varying(200) NOT NULL` – sloupec obsahuje uživatelské jméno.
    5. `body text` – zde je uložen předzpracovaný text zprávy.
    6. sloupce s prefixem `h_` (6 až 27) obsahují předzpracované položky hlavičky zprávy. Suffix vždy vyjadřuje název pole podle RFC 5322.



Obrázek 7.2 Schéma databáze – napojení přidaných tabulek

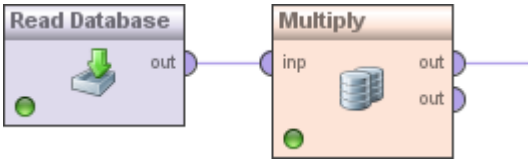
7.4 Model klasifikace

Vytvoření klasifikačního modelu je jednou z hlavních částí celého výsledného systému. Jak bylo popsáno v kapitole 3.1, je nutné nejprve provést fází trénování, kde se vytvoří klasifikační model, ten se otestuje ve fázi testování a potom se již může provádět samotná klasifikace v iterační fázi aplikace vytvořeného modelu.

Klasifikace emailů měla původně probíhat tak, že by se vytvářel model na základě všech částí emailové zprávy, které jsou předzpracované v databázi. Tento postup bohužel není možný, protože nástroj RapidMiner neumožňuje spojovat výsledky, které mají generované sloupce se stejným názvem. Pro toto omezení se používá komponenta pro přejmenování sloupců, ale tento způsob zde nelze použít, protože názvy sloupců jsou názvy složek uživatele a ty nejsou dopředu známy. Vzhledem k uvedeným důvodům se budou klasifikační modely tvořit zvlášť na základě těla zprávy, předmětu, odesílatele a množiny příjemců.

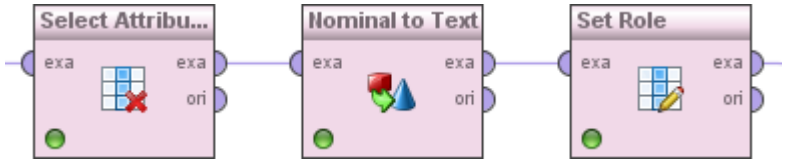
7.4.1 Fáze trénování

Na počátku fáze trénování je nejprve potřeba načíst data, nad kterými se bude provádět trénování klasifikátoru. Obrázek 7.3 nám ukazuje, že je zde použita komponenta *Read Database* pro načtení předzpracovaných emailů z databáze a poté komponenta *Multiply* pro násobení načteného vstupu. Vzhledem k tomu, že zde budou čtyři modely, je tato komponenta nezbytná.



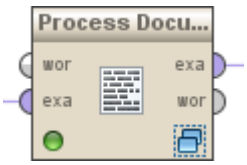
Obrázek 7.3 Získání dat

Dále je potřeba připravit načtená data z databáze. Obrázek 7.4, který je uvedený dále, zobrazuje použité komponenty. *Select Attributes* nejprve vybere položky pro zpracování, konkrétně kombinaci *mailbox_name* a poté jednu z *h_body*, *h_subject*, *h_from* nebo *h_to*. Položky hlavičky se následně převedou na textový typ pomocí komponenty *Nominal to Text*. Na závěr se pomocí komponenty *Set Role* určí, že položka *mailbox_name* je *label*.



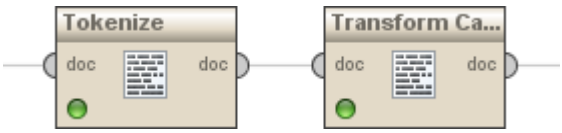
Obrázek 7.4 Příprava dat

Příprava dat je dokončena a nyní se může provést předzpracování. Obrázek 7.5, uvedený níže, zobrazuje komponentu *Process Documents from Data*, která obsahuje předzpracování dat.



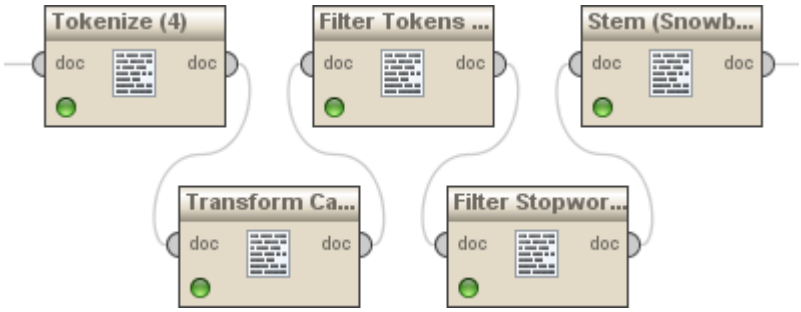
Obrázek 7.5 Zpracování dat

Fáze předzpracování se dělí na dvě části, podle toho jaká data zpracováváme. Obrázek 7.7, který je uvedený níže, zobrazuje předzpracování podle příjemce nebo množiny odesílatelů. V této fázi stačí pouze jednoduché předzpracování, protože vstupem jsou emailové adresy. Nejprve se tedy pomocí komponenty *Tokenize* rozdělí vstup na tokeny, což jsou v našem případě emaily. Poté komponenta *Transform Cases* převede všechna velká písmena na malá.



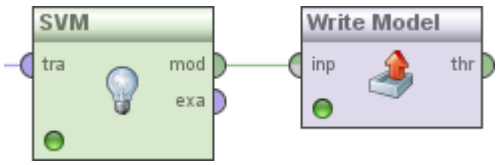
Obrázek 7.6 Předzpracování dat

Obrázek 7.7, uvedený níže, zobrazuje předzpracování dat podle hlavičky nebo těla zprávy. Vzhledem k tomu, že testovací data jsou v anglickém jazyku, je i předzpracování prováděno pro anglický jazyk. Je zde použito více komponent, protože vstup už může obsahovat libovolné věty. Nejprve se vstup rozdělí na tokeny pomocí komponenty *Tokenize* a s pomocí *Transform Cases* se velká písmena převedou na malá. *Filter Tokens* odstraní tokeny, které jsou kratší než čtyři znaky a delší než 25 znaků. Poté jsou odfiltrována anglická stop-slova pomocí komponenty *Filter Stopword*. Na závěr se komponentou *Stem (Snowball)* provede stemming pro anglický jazyk.



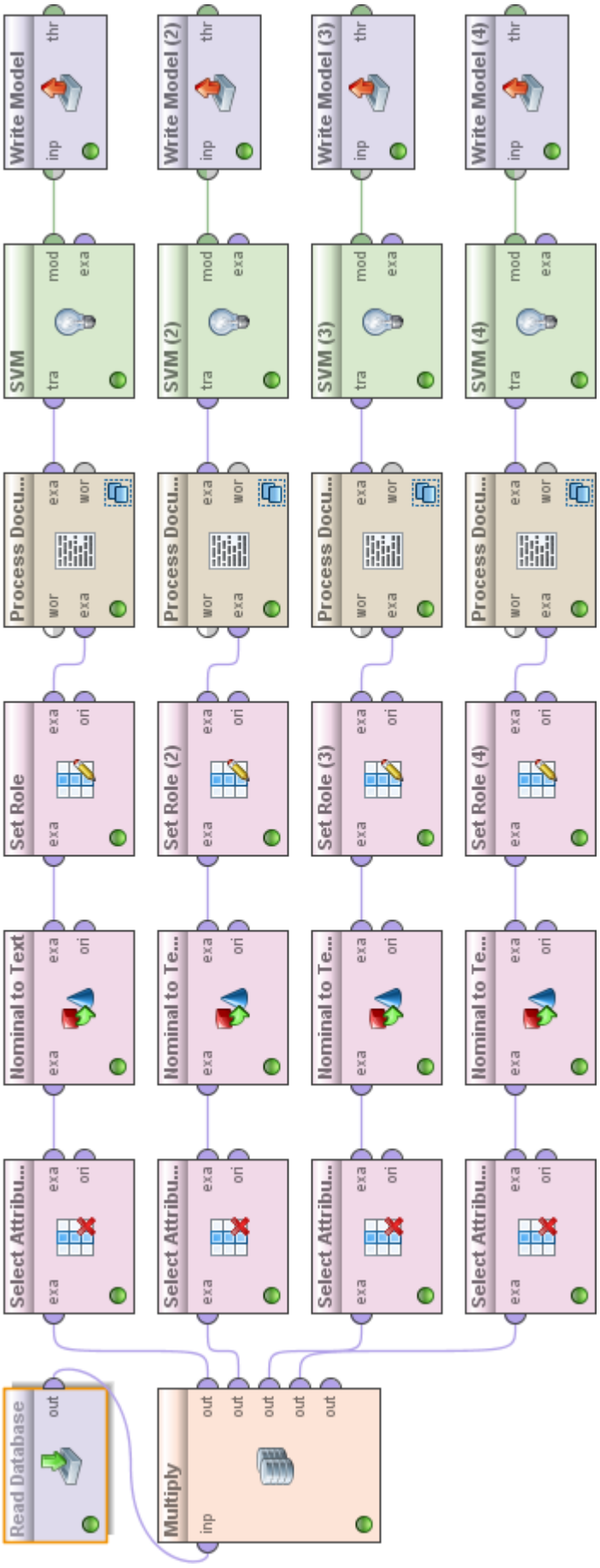
Obrázek 7.7 Předzpracování dat

V závěrečné části trénování se provádí učení klasifikačního modelu. Obrázek 7.8, který je uvedený níže, zobrazuje učení pomocí operátoru *SVM* a následné uložení vytvořeného modelu do souboru pomocí komponenty *Write Model*.



Obrázek 7.8 Učení modelu

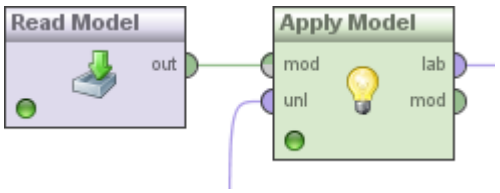
Obrázek 7.9, který je uveden dále, přehledně zobrazuje celý výsledný proces trénování klasifikačního modelu. Na něm již vidíme všechny čtyři části, tedy tvorbu modelu pro tělo zprávy, předmět, odesílatele a příjemce.



Obrázek 7.9 Výsledný proces trénování

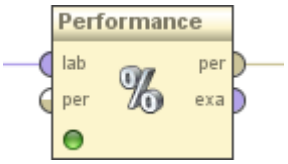
7.4.2 Fáze testování

V této fázi se provádí testování natrénovaného klasifikačního modelu. Nejprve je proto potřeba načíst vytvořený model, což se provádí za pomocí komponenty *Read Model*. Ten je dán na vstup komponenty *Apply Model* společně s testovací sadou emailů. Tato sada by měla být odlišná od množiny emailů, které se použily pro trénování modelu. Obrázek 7.10, uvedený níže, zobrazuje popsanou část použití vytvořeného modelu.



Obrázek 7.10 Použití modelu

Abychom byli schopni zjistit míru úspěšnosti klasifikace, je potřeba použít komponentu *Performance*, která umožňuje procentuální měření úspěšnosti. Obrázek 7.11, uvedený dále, ukazuje měření výkonu. Vstupem jsou emaily, které byly podle modelu klasifikovány a výstupem jsou informace o míře úspěšnosti a další podrobné údaje.



Obrázek 7.11 Měření výkonu

Tabulka 7.1, která je uvedená dále, zobrazuje ukázku výsledků měření výkonosti. Jednotlivé řádky reprezentují předpoklady zařazení do dané složky a sloupce vyjadřují správnost zařazení. Pro každou složku je také vypočtena procentuální úspěšnost klasifikace, uvedená ve sloupci class precision. Celková úspěšnost je zobrazena v záhlaví tabulky. Class recall udává procentové vyjádření správně zařazených emailů.

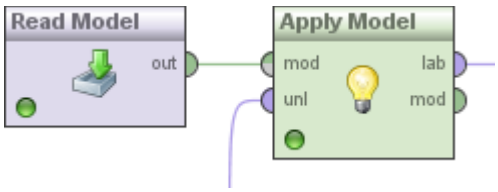
Tabulka 7.1 Výsledky testování

accuracy: 34.08%

	true discussion	true australia	true conferences	true london	true notes	class precision
pred. discussion	154	23	42	56	71	44.51%
pred. australia	1	3	4	7	2	17.65%
pred. conferences	6	10	12	6	7	29.27%
pred. london	12	4	17	6	8	12.77%
pred. notes	33	16	37	23	24	18.05%
class recall	74.76%	5.36%	10.71%	6.12%	21.43%	

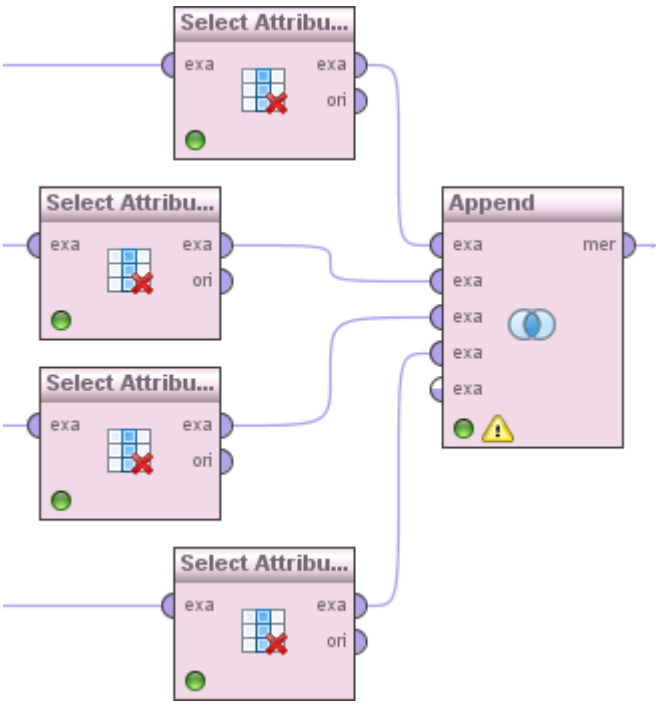
7.4.3 Fáze aplikace

Ve fázi aplikace je potřeba nejprve načíst klasifikační model ze souboru pomocí komponenty *Read Model*. Ten poté slouží jako vstup pro aplikaci modelu na předzpracované emaily uživatele. Použití modelu zajistí komponenta *Apply Model*, do které vstupují emaily ke klasifikaci. Obrázek 7.12, uvedený dále, zobrazuje tyto popsané komponenty a jejich způsob zapojení.



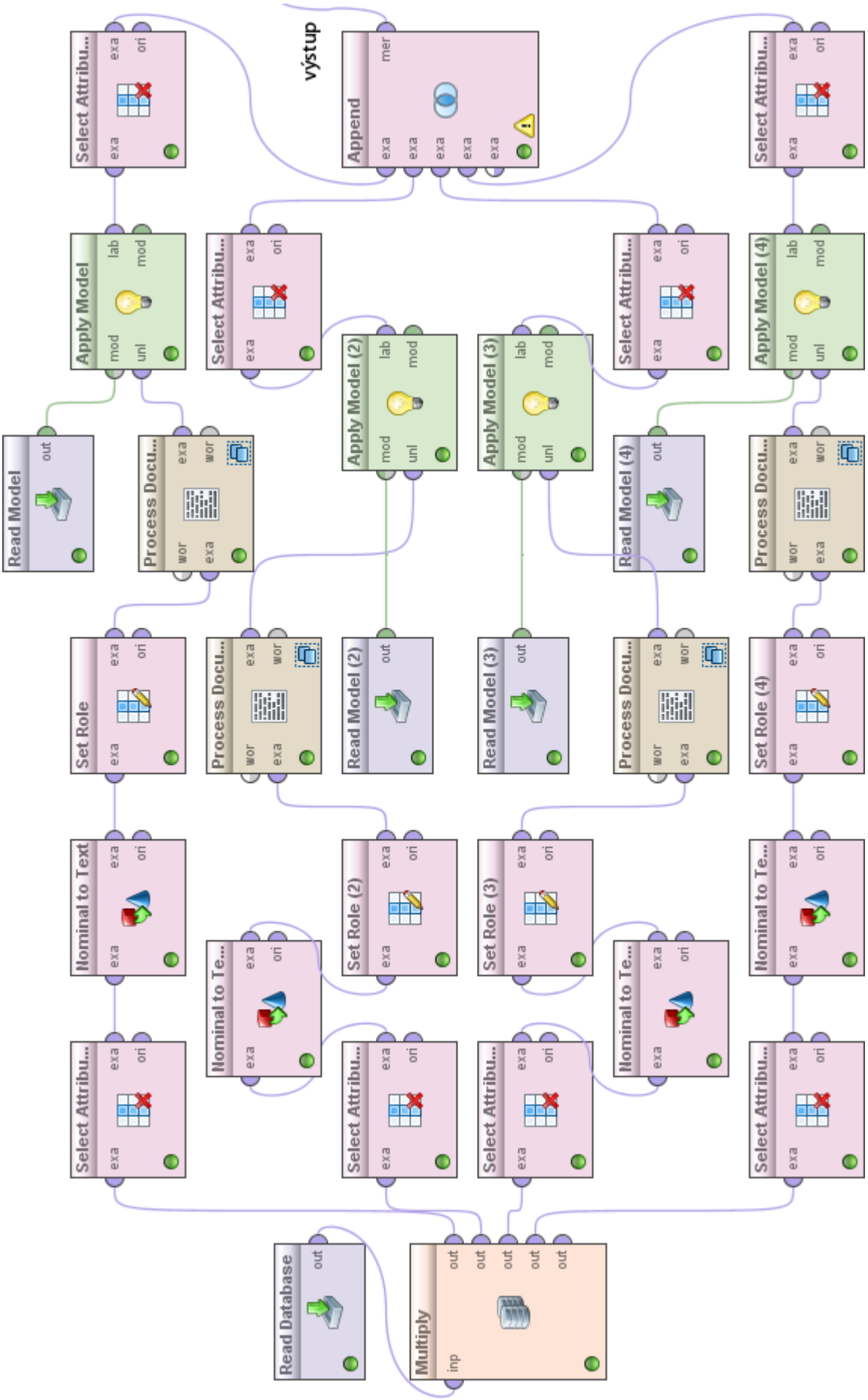
Obrázek 7.12 Použití modelu

Po aplikování modelu jsou pro každý model vráceny výsledky, ze kterých pomocí komponenty *Select Attributes* vybereme pouze id emailu a jméno složky uživatele. Na závěr tyto výsledky spojíme komponentou *Append*. Obrázek 7.13, který je uvedení níže, znázorňuje celý tento popsaný proces.



Obrázek 7.13 Spojení výsledků

Obrázek 7.14, který je uveden dále, přehledně zobrazuje celý výsledný proces klasifikace emailových zpráv. Na něm již vidíme všechny čtyři části, tedy aplikování modelu pro tělo zprávy, předmět, odesílatele a příjemce.



Obrázek 7.14 Výsledný proces klasifikace

7.5 Shrnutí

Na začátku této kapitoly byl uveden diagram komponent, který přehledně shrnuje všechny části provedené implementace. Jedná se o základní komponentu Apache James Server, v níž byly provedeny úpravy pro podporu klasifikace emailů. Ta dále komunikuje s komponentou RapidMiner starající se o fázi učení modelu a klasifikace. Poslední částí je databázová komponenta PostgreSQL pro provádění databázových operací. Dále jsou uvedeny změny provedené v emailovém serveru Apache James Server. Nejprve jsou rozebrány vytvořené mailety pro klasifikaci emailů a import dat. Dále je uveden princip implementace matcherů, což jsou prvky, které určují kdy se má vykonat určitý mailet. Poté je rozepsán zásah do komponenty pro přesun emailů, aby bylo možné vytvořit nový klasifikační model. Následně jsou zmíněny změny v databázi. Je zde zobrazena část schématu databáze, která byla přidána pro potřeby klasifikace. Jedná se o tabulku *data_models* a *data_preprocessing*. Závěrečná část popisuje nejprve tvorbu modelu klasifikace, poté fázi testování a na konec iterativní fázi aplikace. U každé fáze je podrobně rozebrán a zobrazen princip její tvorby. Je zde také ukázán výsledný proces trénování a výsledný proces klasifikace.

8 Experimenty

Součástí tvorby výsledného systému je také provádění nejrůznějších experimentů. Z nich můžeme zjistit zajímavé informace, které nám mohou posloužit k vylepšení nebo úpravě funkčnosti. V této fázi je již implementována klasifikace v emailovém serveru Apache James Server a proto je možné provádět pokusy s tříděním pošty.

8.1 Enron Dataset

Aby bylo možné provádět pokusy a experimenty, jsou potřeba vstupní data, nad kterými by se prováděla klasifikace. Za tímto účelem je použita testovací sada emailů Enron Dataset. Podle [20] byla tato data posbírána a připravena projektem CALO (A Cognitive Assistant that Learns and Organizes). Obsahuje emaily od zhruba 150 uživatelů, které jsou organizovány do složek. Ve většině případů se jedná o zaměstnance managmentu v Enronu. Tato data byla původně zveřejněna a publikována na webu federální agenturou USA nazývanou Federal Energy Regulatory Commission během probíhajícího vyšetřování firmy Enron. Tato testovací sada je uvedena v příloze 1.

8.2 Porovnání klasifikačních metod

Pro účely experimentů byla vybrána jedna sada testovacích emailů z Enron Datasetu, nad kterými se budou provádět experimenty. Nalézá se v ní pět složek – inbox, notes, threads, document a spam. Celkový počet emailů je 35 301, které se skládají z těchto složek:

- **inbox** - 5 095 emailů
- **notes** - 7 194 emailů
- **threads** - 5 550 emailů
- **document** - 9 304 emailů
- **spam** - 8 158 emailů

Aby bylo možné provádět porovnání nad stejnou kolekcí, byly z těchto složek postupně promazávány emaily. Vznikly nám tak další čtyři složky, které mají 25 836, 15 462, 8 031 a 5 240 emailů. Všechny testovací emaily se naimportovaly do databáze a poté byly prováděny experimenty uvedené v této a následujících kapitolách 8.3 a 8.4.

Učení klasifikačního modelu probíhalo nad daty na sudých řádcích v databázi a aplikování klasifikačního modelu se testovalo na lichých řádcích v databázi. Díky tomuto způsobu byla trénovací a testovací data odlišná.

Tabulka 8.1 Výsledky klasifikace metodou SVM (2620 emailů)

accuracy: 18.31%						
	true inbox	true notes	true threads	true document	true spam	class precision
pred. inbox	123	146	46	67	43	28.94%
pred. notes	119	132	255	34	286	15.98%
pred. threads	133	163	123	343	118	13.98%
pred. document	41	37	14	24	14	18.46%
pred. spam	113	53	70	46	78	21.67%
class recall	23.25%	24.86%	24.21%	4.67%	14.47%	

Tabulka 8.2 Výsledky klasifikace metodou NB (2620 emailů)

accuracy: 19.00%						
	true inbox	true notes	true threads	true document	true spam	class precision
pred. inbox	147	139	58	69	56	31.34%
pred. notes	127	139	282	44	280	15.94%
pred. threads	67	115	70	309	64	11.20%
pred. document	12	17	5	16	13	25.40%
pred. spam	176	121	93	76	126	21.28%
class recall	27.79%	26.18%	13.78%	3.11%	23.38%	

Tabulka 8.3 Výsledky klasifikace metodou SVM (4015 emailů)

accuracy: 15.43%						
	true notes	true document	true inbox	true threads	true spam	class precision
pred. notes	320	118	159	138	126	37.17%
pred. document	123	83	76	283	555	7.41%
pred. inbox	150	71	81	32	112	18.16%
pred. threads	176	89	223	97	106	14.04%
pred. spam	161	552	65	82	39	4.34%
class recall	34.41%	9.09%	13.41%	15.35%	4.16%	

Tabulka 8.4 Výsledky klasifikace metodou NB (4015 emailů)

accuracy: 16.75%						
	true notes	true document	true inbox	true threads	true spam	class precision
pred. notes	315	119	193	103	93	38.27%
pred. document	89	71	50	15	43	26.49%
pred. inbox	168	66	87	296	620	7.03%
pred. threads	179	90	169	125	107	18.66%
pred. spam	179	567	105	93	75	7.36%
class recall	33.87%	7.78%	14.40%	19.78%	8.00%	

Tabulka 8.5 Výsledky klasifikace metodou SVM (7731 emailů)

accuracy: 15.68%

	true document	true notes	true spam	true inbox	true threads	class precision
pred. document	453	507	322	206	177	27.21%
pred. notes	334	426	266	138	72	34.47%
pred. spam	155	308	202	76	305	19.31%
pred. inbox	148	120	1111	64	11	4.40%
pred. threads	1618	292	234	120	67	2.87%
class recall	16.73%	25.77%	9.46%	10.60%	10.60%	

Tabulka 8.6 Výsledky klasifikace metodou NB (7731 emailů)

accuracy: 14.15%

	true document	true notes	true spam	true inbox	true threads	class precision
pred. document	253	270	1101	126	82	13.81%
pred. notes	522	437	417	209	100	25.93%
pred. spam	346	585	293	131	360	17.08%
pred. inbox	156	123	140	52	31	10.36%
pred. threads	1431	238	184	86	59	2.95%
class recall	9.34%	26.44%	13.72%	8.61%	9.34%	

Tabulka 8.7 Výsledky klasifikace metodou SVM (12918 emailů)

accuracy: 30.85%

	true document	true notes	true inbox	true threads	true spam	class precision
pred. document	461	636	314	149	372	23.86%
pred. notes	2163	626	398	279	601	15.39%
pred. inbox	96	72	76	39	84	20.71%
pred. threads	481	768	365	1135	388	36.18%
pred. spam	505	549	448	227	1688	49.40%
class recall	12.44%	23.61%	4.75%	62.06%	53.88%	

Tabulka 8.8 Výsledky klasifikace metodou NB (12918 emailů)

accuracy: 19.68%

	true document	true notes	true inbox	true threads	true spam	class precision
pred. document	493	611	378	189	451	23.23%
pred. notes	2192	527	500	320	522	12.98%
pred. inbox	285	291	156	46	1683	6.34%
pred. threads	256	680	213	1101	211	44.74%
pred. spam	480	542	354	173	266	14.66%
class recall	13.30%	19.88%	9.74%	60.20%	8.49%	

Tabulka 8.9 Výsledky klasifikace metodou SVM (17650 emailů)

accuracy: 12.84%						
	true document	true inbox	true notes	true spam	true threads	class precision
pred. document	434	418	675	455	323	18.83%
pred. inbox	48	45	183	22	1275	2.86%
pred. notes	654	535	778	480	429	27.05%
pred. spam	2658	718	813	520	258	10.47%
pred. threads	858	832	1148	2602	490	8.26%
class recall	9.33%	1.77%	21.63%	12.75%	17.66%	

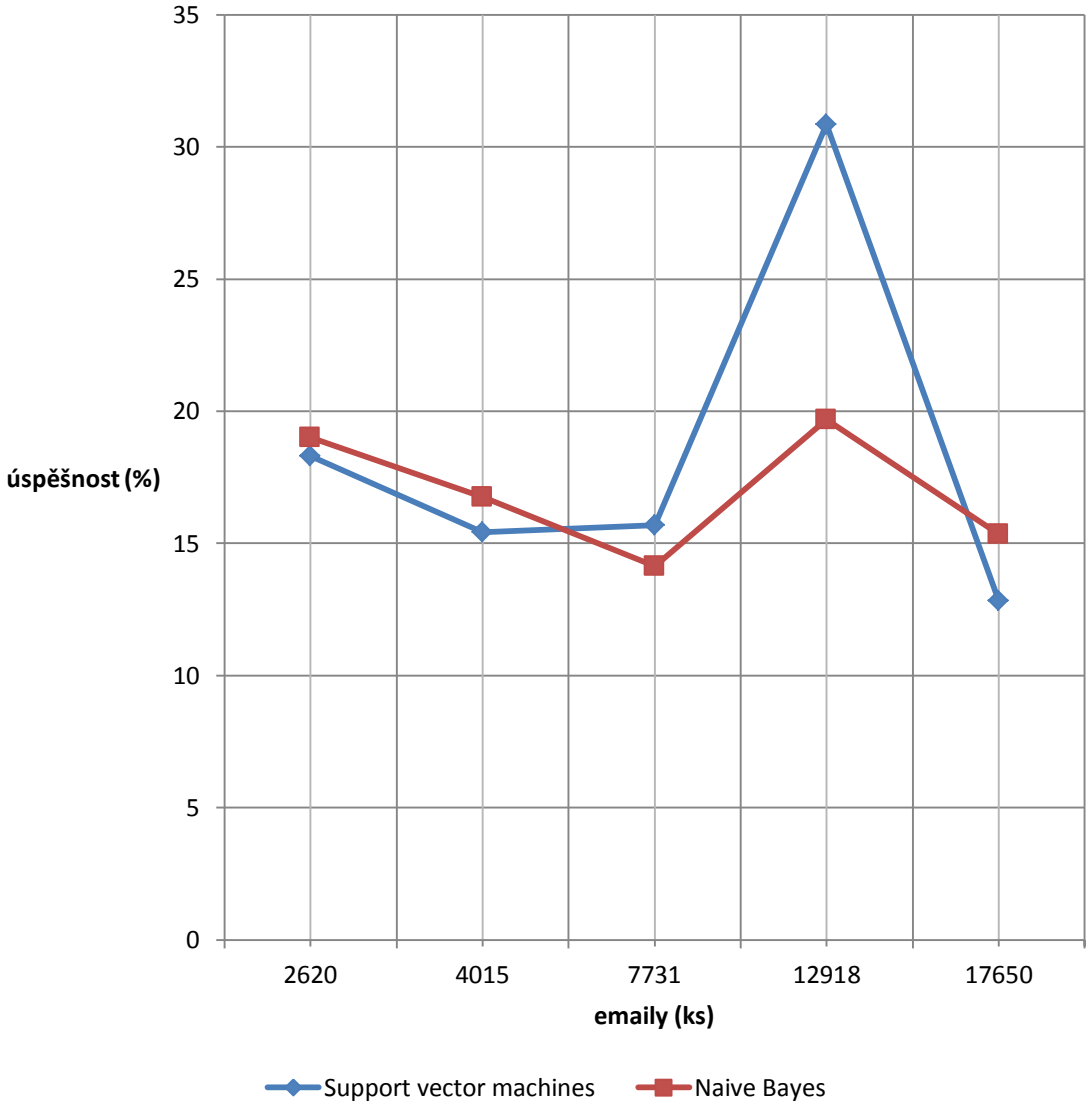
Tabulka 8.10 Výsledky klasifikace metodou NB (17650 emailů)

accuracy: 15.34%						
	true document	true inbox	true notes	true spam	true threads	class precision
pred. document	472	443	402	2360	1468	9.17%
pred. inbox	272	264	477	206	144	19.37%
pred. notes	638	589	980	508	400	31.46%
pred. spam	2585	712	872	629	400	12.10%
pred. threads	685	540	866	376	363	12.83%
class recall	10.15%	10.36%	27.24%	15.42%	13.08%	

8.3 Vliv dat na úspěšnost klasifikace

Graf 8.1, který je uveden níže, vychází z výsledků jednotlivých metod uvedených v kapitole 8.2. Na grafu vidíme, že u metody SVM dochází nejprve ke kolísání míry úspěšnosti. Je to dáno tím, že se projevuje vzrůstající počet vstupních emailů, který zpřesňuje úspěšnost. Dále dochází k růstu úspěšnosti až na hodnotu 30,85%, kde je vrchol. Nyní jsou již vstupní data tak rozsáhlá, že dochází k přetrénování klasifikátoru a proto se zvyšuje míra špatné klasifikace. Podobná situace nastává u metody NB, ale s mírnějším průběhem. Zde je zajímavé, že tato metoda má úspěšnost 19% na vzorku 2620 emailů a 19.68% na vzorku 12918 emailů. Z experimentu vyplývá, že SVM sice může dosáhnout lepších výsledků, ale na druhou stranu je více ovlivněn vstupními daty.

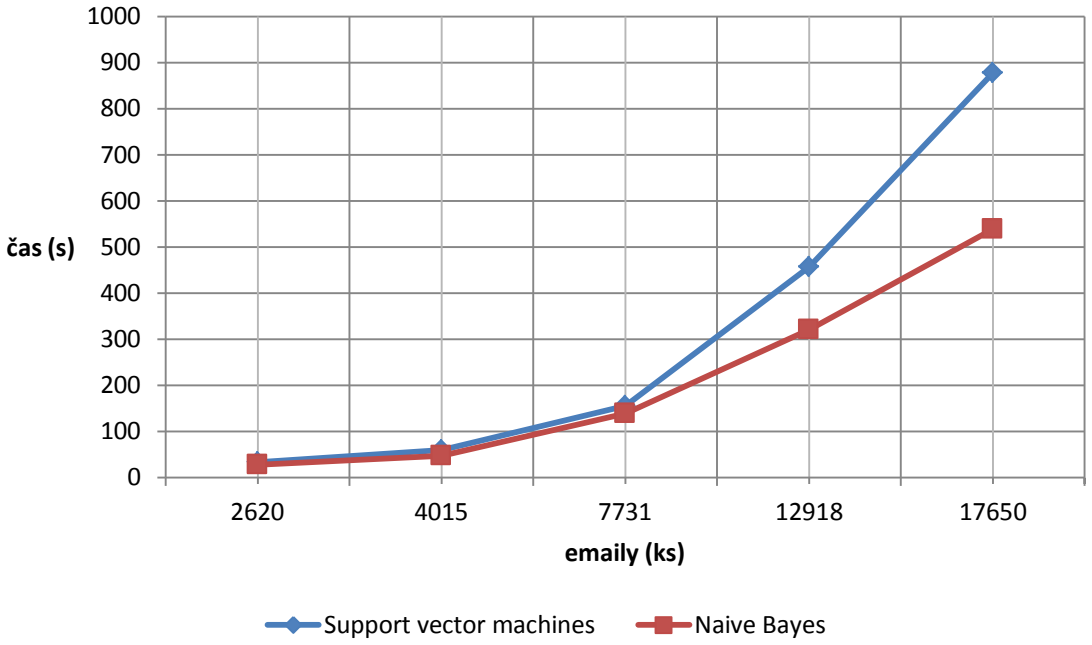
Graf 8.1 Vliv dat na úspěšnost klasifikace



8.4 Doba učení a aplikace modelu

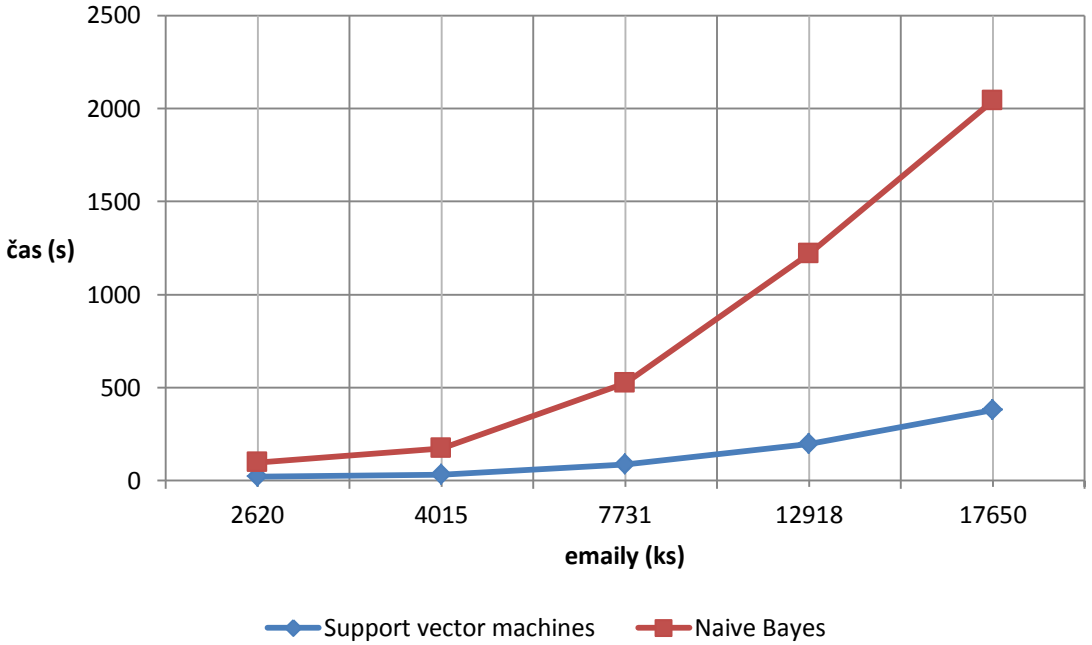
Graf 8.2, uvedený níže, zobrazuje dobu učení modelu vzhledem k velikosti vstupních dat. Zpočátku je čas u obou metod podobný, ale zhruba na úrovni 8 tisíc emailů začíná metoda SVM vyžadovat více času na učení. Na úrovni 18 tisíc emailů je již tento čas o více jako polovinu větší než u metody NB.

Graf 8.2 Porovnání doby učení modelu vzhledem ke vstupním datům



Graf 8.3 zobrazuje dobu potřebnou pro aplikování vytvořeného klasifikačního modelu na vzorek vstupních emailů. Již od počátku je vidět znatelný rozdíl mezi metodou SVM a NB. Zatímco čas u metody SVM roste v závislosti na datech pomalu, u metody NB prudce stoupá již od počátku.

Graf 8.3 Porovnání doby aplikování modelu vzhledem ke vstupním datům



8.5 Shrnutí

Tato kapitola se zabývala experimenty s testovací sadou emailů Enron Dataset a demonstračním systémem pro klasifikaci příchozí pošty. Nejprve bylo provedeno srovnání klasifikačních metod Support vector machines a Naive Bayes z pohledu úspěšnosti klasifikace. Poté byl ukázán vliv vstupních dat na úspěšnost klasifikace a na závěr bylo uvedeno porovnání doby potřebné pro učení modelu a také doby potřebné pro aplikaci klasifikačního modelu.

Výsledky klasifikace nejsou uspokojivé, což může ovlivnit několik faktorů. Jedná se o vstupní množinu dat, která nemusí být ideální, ale reprezentuje reálný vzorek emailů. Dalším faktorem je proces tvorby modelu klasifikace, což je obtížný úkol, který vyžaduje řadu pokusů a optimalizací. Právě tento model je klíčovým prvkem, který může podstatně vylepšit výsledky.

9 Závěr

Hlavním cílem této diplomové práce je vytvoření demonstračního systému, který umožňuje klasifikovat příchozí poštu, a následně provádět experimenty nad množinou testovacích dat. Součástí je také zhodnocení výsledného systému, použitých nástrojů a porovnání použitých metod.

K dosažení tohoto cíle bylo nutné nejdříve prostudovat obecnou problematiku získávání znalostí z databází a následně se zaměřit na klasifikaci textu. Poté bylo zapotřebí seznámit se s obecnými principy emailové komunikace s důrazem na komunikační protokol IMAP. Tyto teoretické znalosti posloužily jako výchozí bod při návrhu výsledného systému.

Ve fázi návrhu byl vytvořen základní princip klasifikace emailů a byla provedena volba a analýza vhodných nástrojů. Jedná se o emailový server Apache James Server, dolovací nástroj RapidMiner a databázový systém PostgreSQL. Následovala implementační fáze, ve které se nejprve upravil a poté rozšířil emailový server tak, aby umožňoval třídění příchozí pošty. Pro samotnou klasifikaci byl využit nástroj RapidMiner. V něm byl vytvořen proces klasifikace umožňující tvorbu klasifikačního modelu, a proces aplikace pro jeho použití. Dále bylo rozšířeno databázové schéma, aby podporovalo uložení předzpracovaných dat a informací o klasifikačním modelu uživatele. Na závěr byly provedeny experimenty s metodami Support vector machines (SVM) a Naive Bayes (NB) spolu s jejich porovnáním.

Výhodou použitého emailového serveru je dobrá podpora práce s emaily. To je zajištěno díky tzv. mailetům, což jsou komponenty ve kterých je umístěno veškeré zpracování příchozí pošty. Ve své podstatě se jedná o modulární systém, který umožňuje snadno přidávat nebo odebírat požadovanou funkčnost. Nevýhodou tohoto serveru je množství chyb a nedostatečná dokumentace.

Dolovací nástroj RapidMiner má výhodu v tom, že umožňuje vytvořit vlastní proces v desktopové aplikaci a následně ho použít ve vlastní implementaci. Jeho výhodou je rozsáhlá podpora práce s textem a nejrůznější komponenty pro optimalizace nebo měření výkonu. Nevýhoda spočívá ve stručné dokumentaci, omezení při zpracování dat nebo nemožnosti provádět spojování generovaných výsledků.

Funkčnost použitých nástrojů byla demonstrována na testovacím vzorku emailů Enron Dataset. Z experimentů je vidět, že průměrná úspěšnost klasifikace metodou SVM je 18,6% a metodou NB je 16,9%. Tyto výsledky nejsou uspokojivé, přestože byly prováděny nad vzorkem emailů, který čítal od 2 do 17 tisíc zpráv. Zajímavé dále je, že úspěšnost metody SVM nad vzorkem 13 tisíc emailů byla 30,85%, ale nad větší množinou 18 tisíc emailů byla pouhých 12,84%. Tento propad byl také u metody NB, kde nebyl tolik patrný, kdy u stejného vzorku dat se jednalo o úspěšnost 19,68% a 15,34%. Vzhledem k principům obou metod bych předpokládal, že SVM bude výrazně lepší než NB. Tyto výsledky jsou ale ve velké míře ovlivněny procesem tvorby modelu obsahujícím nejrůznější parametry. Při jejich optimalizaci jsem se snažil o dosažení co nejlepších výsledků, ale vzhledem k experimentům se stále nejedná o nejlepší řešení.

Možné rozšíření práce do budoucna se dá rozdělit do několika oblastí, podle použitých nástrojů. U emailového serveru Apache James Server by bylo dobré přidat podporu stahování pošty ze zadaných schránek. Server by tímto umožňoval stahování pošty z několika uživatelských účtů do jedné lokální schránky, ve které by se prováděla klasifikace. U nástroje RapidMiner by bylo dobré dále experimentovat s procesem tvorby modelu, aby se zvýšila úspěšnost třídění příchozí pošty. Toho by se dalo dosáhnout tak, že by se samotná klasifikace prováděla nad všemi položkami emailové zprávy. RapidMiner tento způsob neumožňuje, proto by se musel buď použít jiný nástroj, nebo implementovat obslužný prvek mezi Apache James Serverem a RapidMinerem.

Literatura

[1] ZENDULKA, Jaroslav, Vladimír BARTÍK, Roman LUKÁŠ a Ivana RUDOLFOVÁ. *Získávání znalostí z databází*. Brno, 2009. Studijní opora. FIT VUT v Brně.

[2] BERKA, Petr. *Dobývání znalostí z databází*. 1. vydání. Praha: Academia, 2003, 366 s. ISBN 80-200-1062-9.

[3] HAN, Jiawei a Micheline KAMBER. *Data Mining: Concepts and Techniques*. Second Edition. San Francisco: Morgan Kaufmann Publishers, 2006, 770 s. ISBN 1-55860-901-6.

[4] CHMELAŘ, Petr. *Získávání znalostí z multimediálních databází*. Brno, 2007. Elektronický učební text. FIT VUT v Brně.

[5] CHMELAŘ, Petr a Lukáš STRYKA. *Multimediální databáze*. Brno, 2007. Elektronický učební text. FIT VUT v Brně.

[6] FELDMAN, Ronen a James SANGER. *The Text Mining Handbook: Advanced Approaches in Analyzing Unstructured Data*. New York: Cambridge University Press, 2007, 410 s. ISBN 0-521-83657-3.

[7] BERRY, Michael W. (editor) a Jacob KOGAN (editor). *Text Mining: Applications and Theory*. Chichester: Wiley, 2010, 207 s. ISBN 978-0-470-74982-1.

[8] ROBERTSON, Stephen. Understanding Inverse Document Frequency: On theoretical arguments for IDF. *Journal of Documentation*. 2004, Number 5, 503–520. Dostupné z: http://www.soi.city.ac.uk/~ser/idfpapers/Robertson_idf_JDoc.pdf

[9] CROCKER, Dave. *Email History*. [online]. [cit. 2012-01-01]. Dostupné z: http://www.livinginternet.com/e/ei.htm

[10] HALSALL, Fred. *Computer Networking and the Internet*. Fifth edition. Edinburg: Addison-Wesley, 2005, 803 s. ISBN 0-321-26358-8.

[11] RFC 5322. *Internet Message Format.* Říjen 2008. Dostupné z: http://tools.ietf.org/html/rfc5322

[12] TANENBAUM, Andrew S. *Computer Networks*. 4th ed. New Jersey: Prentice Hall, 2003, 891 s. ISBN 0-13-066102-3.

[13] RFC 5321. *Simple Mail Transfer Protocol.* Říjen 2008. Dostupné z: http://tools.ietf.org/html/rfc5321

[14] RFC 1939. *Post Office Protocol – Version 3.* Květen 1996. Dostupné z: http://tools.ietf.org/html/rfc1939

[15] RFC 3501. *INTERNET MESSAGE ACCESS PROTOCOL – VERSION 4rev1*. Březen 2003. Dostupné z: http://tools.ietf.org/html/rfc3501

[16] THE APACHE SOFTWARE FOUNDATION. James Server [online]. 2012-04-17 [cit. 2012-05-01]. Dostupné z: http://james.apache.org/server/3/

[17] *PostgreSQL: wiki* [online]. [cit. 2012-05-01]. Dostupné z: http://postgres.cz/wiki/PostgreSQL

[18] RAPID-I GMBH. *Rapid-i: report the future* [online]. Dortmund [cit. 2012-05-01]. Dostupné z: http://www.rapid-i.com

[19] RAPID-I GMBH. *RapidMiner 5.0: Manual* [online]. Dortmund, 2010 [cit. 2012-05-01]. Dostupné z: http://rapid-i.com/content/view/26/84/lang,en/

[20] Enron Email Dataset. *School of Computer Science, Carnegie Mellon* [online]. Fri Aug 21 2009 [cit. 2012-05-01]. Dostupné z: http://www.cs.cmu.edu/~enron/

Seznam příloh

Příloha 1. DVD obsahující složky:

- **Apache James Server** – obsahuje zdrojové kódy, potřebné knihovny a návod na instalaci.
- **Enron Dataset** – testovací sada emailů Enron.
- **RapidMiner** – soubory obsahující vytvořené klasifikační modely.
- **Schema databaze - rozsirene** – rozšířené schéma databáze.
- **Schema databaze - zakladni** – základní schéma databáze.
- **Technicka zprava** – soubory technické zprávy.