VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV INFORMAČNÍCH SYSTÉMŮ
DEPARTMENT OF INFORMATION SYSTEMS

ANALÝZA A ZÍSKÁVÁNÍ INFORMACÍ ZE SOUBORU DOKUMENTŮ SPOJENÝCH DO JEDNOHO CELKU

ANALYSIS AND DATA EXTRACTION FROM A SET OF DOCUMENTS MERGED TOGETHER

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE **Bc. JORDÁN JAROLÍM**
AUTHOR

VEDOUCÍ PRÁCE **doc. RNDr. JITKA KRESLÍKOVÁ, CSc.**
SUPERVISOR

BRNO 2018

Zadání diplomové práce/20912/2017/xjarol03

Vysoké učení technické v Brně - Fakulta informačních technologií

Ústav informačních systémů Akademický rok 2017/2018

Zadání diplomové práce

Řešitel: **Jarolím Jordán, Bc.**
Obor: Bezpečnost informačních technologií
Téma: **Analýza a získávání informací ze souboru dokumentů spojených do jednoho celku**
Analysis and Data Extraction from a Set of Documents Merged Together
Kategorie: Zpracování řeči a přirozeného jazyka

Pokyny:

1. Prostudujte existující metody pro automatizované rozdělování dokumentů do dílčích částí dle obsahu a pro získávání relevantních informací z dokumentů.
2. Specifikujte požadavky na aplikaci pro efektivní rozdělování rozsáhlých dokumentů.
3. Navrhněte algoritmus pro automatické rozdělování několika spojených dokumentů.
4. Zvolte vhodné vývojové prostředí a implementujte prototyp navrženého systému. Lze použít existující nástroje pro extrakci relevantních dat z jednotlivých dokumentů. Jednotlivé dokumenty zobrazte na časové ose umožňující uživatelům editovat případné odchylky.
5. Na vzorku dat, vybraném po dohodě s vedoucí, ověřte použitelnost vytvořeného prototypu nástroje a otestujte jeho kvalitu vůči získanému vzorku referenčních hodnot.
6. Zhodnoťte dosažené výsledky a diskutujte další možná rozšíření.

Literatura:

- BERRY, Michael a Jacob KOGAN, ed. *Text Mining: Applications and Theory:* John Wiley & Sons, 2010. ISBN 978-0-470-74982-1.
- CHERIET, Mohamed, Nawwaf KHARMA, Cheng-Lin LIU a Ching SUEN. *Character Recognitions Systems: A Guide for Students and Practitioners:* John Wiley & Sons, 2007. ISBN 978-0-471-41570-1.
- A comprehensive survey of mostly textual document segmentation algorithms since 2008. Pattern Recognition. 2017, 64, 1-14. DOI: 10.1016/j.patcog.2016.10.023. ISSN 00313203

Při obhajobě semestrální části projektu je požadováno:

- Splnění bodů zadání 1 až 3.

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 dřívějších projektů (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: **Kreslíková Jitka, doc. RNDr., CSc.**, UIFS FIT VUT
Datum zadání: 1. listopadu 2017
Datum odevzdání: 23. května 2018

L.S.

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

doc. Dr. Ing. Dušan Kolář
vedoucí ústavu

Abstrakt

Tato práce se zabývá získáváním relevantních informací z dokumentů, automatizovaným rozdělováním vícero dokumentů spojených do jednoho celku a tvorbou nástroje, který umožňuje získání relevantních informací z dokumentů a jejich automatizované rozdělení. Jsou diskutovány především metody pro získání textových dat ze skenovaných dokumentů, rozpoznávání pojmenovaných entit, shlukování dokumentů, jejich podpůrné algoritmy a jsou popisovány metriky sloužící pro automatizované rozdělování dokumentů. Dále je vysvětlen algoritmus implementovaného prototypu daného systému, jsou popsány použité nástroje a techniky a je evaluována jeho úspěšnost. Nakonec jsou diskutována možná rozšíření a budoucí rozvoj této práce.

Abstract

This thesis deals with mining of relevant information from documents and automatic splitting of multiple documents merged together. Moreover, it describes the design and implementation of software for data mining from documents and for automatic splitting of multiple documents. Methods for acquiring textual data from scanned documents, named entity recognition, document clustering, their supportive algorithms and metrics for automatic splitting of documents are described in this thesis. Furthermore, an algorithm of implemented software is explained and tools and techniques used by this software are described. Lastly, the success rate of the implemented software is evaluated. In conclusion, possible extensions and further development of this thesis are discussed at the end.

Klíčová slova

Dokument, spojené dokumenty, rozdělování, automatické rozdělování dokumentů, získávání informací, rozpoznávání pojmenovaných entit, RAKE, SIFT points, histogram, K-means, metrické prostory, vektorové prostory, euklidova vzdálenost, cosinova podobnost, shlukování dokumentů

Keywords

Document, merged documents, splitting, automatic document splitting, data mining, named entity recognition, RAKE, SIFT points, histogram, K-means, metric spaces, vector spaces, euclid distance, cosine similarity, document clustering

Citace

JAROLÍM, Jordán. *Analýza a získávání informací ze souboru dokumentů spojených do jednoho celku.* Brno, 2018. Diplomová práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce doc. RNDr. Jitka Kreslíková, CSc.

Analýza a získávání informací ze souboru dokumentů spojených do jednoho celku

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně pod vedením paní doc. RNDr. Jitky Kreslíkové, CSc. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Jordán Jarolím
21. května 2018

Poděkování

Děkuji paní doc. RNDr. Jitce Kreslíkové, CSc. za odborné vedení této diplomové práce a za její cenné rady a materiály, které mi poskytla. Dále děkuji své rodině za podporu a pomoc po celou dobu studia.

Obsah

Kapitola 1

Úvod

Přestože je v dnešní době valná většina dokumentů uložena v elektronické podobě, obsahují stále velké množství nestrukturovaných, počítačově těžko zpracovatelných dat. To je způsobeno například jazykovou variabilitou. Navíc stále existují důvody k udržování jejich fyzické podoby, tedy vytištění na papír. Může se jednat o subjektivní přehlednost tištěných materiálů, efektivnější vepisování poznámek, či nutnost autentických podpisů a další požadavky. To jde však proti obecnému trendu efektivního a rychlého vyhledávání informací, kategorizace dokumentů a v případě fyzických dokumentů také proti sdílení mezi různými lidmi, či uzly na síti, pro které je nutná digitální reprezentace dokumentu. Z tohoto konfliktu požadavků tedy plyne zásadní potřeba dovednosti zpracovávat dokumenty obsahující nestrukturovaná data, či vytištěné (fyzické) dokumenty stejně efektivně, jako jejich elektronické strukturované vzory, nebo obecné potlačení přirozeně jazykové (nestrukturované) i fyzické (tištěné) reprezentace. Jen velmi těžko lze však donutit uživatele, aby se vzdali svých návyků a začali používat výhradně elektronické zdroje, potlačili své jazykové dovednosti a začali komunikovat pouze předdefinovanou strukturovanou formou. Z tohoto důvodu je pro efektivní práci s dokumenty nutné zpracovávat dokumenty tvořené lidmi, jako by obsahovaly čistě strukturovaná data a fyzické dokumenty alespoň z části tak, jako by se jednalo o jejich digitální formu.

V této práci jsou popsány algoritmy pro získávání relevantních dat a entit nejen z elektronických dokumentů obsahující textová nestrukturovaná data, ale i ze skenovaných dokumentů, u kterých je prvním krokem texty vůbec získat. Dále se zabývá specifickým případem zpracování skenovaných dokumentů, konkrétněji je zaměřena na automatizované rozdělování vícero dokumentů, které byly spojeny skenováním v jeden celek, na logicky související části na základě informací, které lze z obrazové (skenované) reprezentace získat. Lze si představit sken smlouvy a všech souvisejících příloh, jako například technické dokumentace. V takovém případě je pro kategorizaci žádoucí rozdělit výsledný sken na text samotné smlouvy, soubory technické dokumentace a další případné přílohy. K takto rozděleným dokumentům lze pak samozřejmě přiřadit relevantní metadata, na základě kterých je možná kategorizace a případně vyhledávání. Práce navíc popisuje další možné metriky a metody pro zjištění podobnosti či rozdílnosti částí skenovaného dokumentu, které mohou sloužit jako další vodítko k segmentaci dokumentu.

V první části práce (kapitola 2) jsou popsány algoritmy pro získání textových informací z obrazů (skenů) dokumentů - konkrétněji *Optical Character Recognition*. Tato kapitola rozvíjí 1. část 2. kapitoly semestrálního projektu. Druhá část práce (kapitola 3) se zabývá získáváním relevantních dat z dokumentů, které mohou později sloužit pro jejich kategorizaci a efektivní vyhledávání. Je přirozeným pokračováním stejnojmenné 2. části 1. kapitoly

semestrálního projektu. Další kapitola 4 je věnována metodám zjišťování podobnosti dokumentů. Zde je kladen důraz na jejich obecné principy tak, aby mohly být využity pro cílené automatizované rozdělování dokumentů. Jedna z podkapitol se věnuje také problematice shlukování dokumentů, která byla zevrubně popsána v semestrálním projektu v kapitole 3. Poté (kapitola 5) je důkladně rozebrán a popsán implementovaný prototyp aplikace pro automatizované rozdělování vícero dokumentů spojených do jednoho celku a získávání relevantních informací a je zhodnocena úspěšnost vytvořeného systému. Závěrem (kapitola 6) jsou shrnuty dosažené výsledky této práce a je diskutován další možný rozvoj.

Kapitola 2

OCR

Tato kapitola se zabývá především získáváním textu z dokumentů, jež neobsahují žádná textová a jednoduše zpracovatelná data (kromě metadat). Například se tedy může jednat o skeny tištených textových dokumentů, novinových článků obsahujících obrázky, emailů, lékařských zpráv a dalších. Obecně se disciplína zabývající se problematikou zpracování dokumentů, jež jsou reprezentovány obrazovou formou, nazývá *Document Image Analysis* a jejím cílem je rozpoznání textu a grafických komponent a extrakce požadovaných informací z obrazů/skenů dokumentů takovým způsobem, jakým by to bylo přirozené pro člověka. Existuje několik typů analýzy - jsou to zpracování textových komponent, zpracování obrazových komponent a dále zpracování jiných grafických komponent dokumentu, jako například grafy a diagramy [24].

Z výše uvedeného vyplývá, že v této kapitole je popisována především analýza textových komponent, pro kterou je v případě obrazových dat ve skenovaných dokumentech hlavní rozpoznání jednotlivých písmen - *OCR - Optical Character Recognition* a na to navazující rozpoznávání slov, vět, odstavců. Po získání textové reprezentace dokumentu je možno získat entity jako osoby, organizace, datumy a další. Dle těchto entit se dá odhadnout například téma části dokumentu. To může sloužit jako markant pro kategorizaci a pro porovnání se zbytkem dokumentu. Zmíněné grafické a obrazové komponenty jsou zpracovávány pouze z pohledu podobnosti na více stránkách. Toto je však popsáno v následujících kapitolách, např. 4.

Je však důležité také zmínit, že některé metody a názvy popisované v této kapitole slouží také jako malý slovník pojmů, na který bude odkazováno v dalších částech věnujících se naprosto jiným tématům. To z důvodu, že většina technik a pojmů spjatých s OCR má interdisciplinární přesah a mají širokou škálu využití.

2.1 Základní principy OCR

OCR neboli *Optical Character Recognition* je disciplína zabývající se získáváním textových dat rozpoznáváním písmen, slov a vět z obrazové reprezentace dokumentů. Pro porozumění moderním technikám v OCR je nutné se lehce podívat do historie. První náznaky automatického rozpoznávání znaků se datují do 50. let minulého století. Úspěšnost tehdejších systémů však nebyla nikterak přesvědčivá. To bylo způsobeno především nekvalitním tiskem, velkou variabilitou fontů a používanými metodami pro rozpoznávání znaků. Dřívější systémy byly založeny především na tzv. *template matching*, v některé literatuře také jako *matrix matching* [31], který měl problémy především se zmíněnou variabilitou. Celý algorit-

mus vychází z rozdělení vstupního obrazu obsahujícího neznámý znak na matici o dohodnuté velikosti. Každá část matice je pak prahováním převedena na černou/bílou. Odpovídající pixely jsou pak porovnány v každém dostupném vzoru. Počet pixelů, jejichž barva je stejná jako barva ve vzoru, je zaznamenán, vzor s nejvyšším skóre vyhrává a je tedy určen jako rozpoznaný znak. Jako optimalizace mohou být například některé pixely ve vzoru úplně ignorovány, pokud nabývají ve všech vzorech stejných hodnot. Je zřejmé, že tento způsob neřeší rozdílné velikosti, natočení, či jiné deformace znaků [31].

V pozdější historii jsme mohli být svědky vzniku a standardizace speciálních fontů určených právě pro OCR, v 70. letech minulého století např. OCRA, OCRB. Tyto standardy velmi zvedly úspěšnost OCR systémů. Stále se však jedná o systémy založené na *template matching*, které nelze z výše uvedených důvodů úspěšně použít například pro ručně psané texty [12].

S těmi si umí více méně poradit až metody založené na strukturální analýze znaků. Dá se říci, že struktura jakéhokoliv znaku může být rozdělena na menší podčásti. U každé takovéto podčásti je možno identifikovat různé vlastnosti a vztahy mezi těmito vlastnostmi a vztahy mezi jednotlivými podčástmi [22]. Takové vlastnosti, které jsou použitelné například i pro OCR se obecně nazývají *features*, a jedním ze stavebních kamenů dnešních OCR systémů je právě extrakce takových vlastností - *feature extraction*. Tento obecný pojem úzce souvisí nejen s *pattern recognition*, ale také s rozpoznáváním obrazu a se strojovým učením. Z pohledu OCR se *feature extraction* snaží získat o znaku informace vyššího řádu než pouze jeho bitmapu. Tím se stává OCR nezávislé na zvoleném typu písma, či jeho deformacích. Nevýhodou takového přístupu je jejich časová náročnost, která je vyšší než v případě *template matching* [31]. Tato práce uvádí v následující kapitole některé z obecných vlastností (*features*), které mohou být využity nejen pro rozpoznávání znaků při OCR. Rozhodně však nejsou v této práci uvedeny všechny možné *features* a uvedený zevrubný přehled slouží spíše pro lepší představu a pochopení problematiky. Dnešní systémy mají tedy v podstatě tři fáze zpracování dokumentů, jsou jimi [12, 30]:

1. Předzpracování sloužící k vylepšení vstupního obrazu a případné selekci zájmových oblastí.

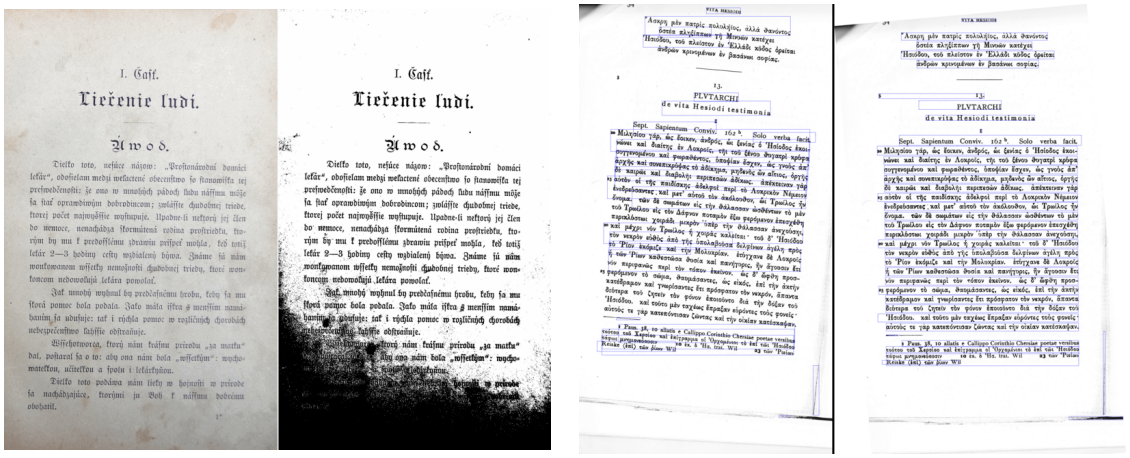
2. Extrakce vlastností zatím neznámých znaků.

3. Klasifikace zpracovávající vektory vlastností pro identifikaci znaků a slov.

2.1.1 Příprava dokumentu pro získání textových dat

Přestože si moderní systémy umí částečně poradit s různými deformacemi při detekci znaků, jako je například natočení stránky [30], stále je pro co nejlepší výsledky vhodné provádět tzv. preprocessing. Mezi klasické obrazové úpravy patří vylepšení lokálních i globálních hodnot histogramu, lokální či globální binarizaci nastavením vhodného prahu - převod do černobílé (obr. 2.1a), oprava natočení (obr. 2.1b), odstraňování šumu, odstranění okrajů a další. Většinou se jedná o obecné obrazové úpravy, které mohou být individuální pro každý dokument, současně je však spektrum jejich použití daleko širší než pouhé OCR.

Po uvedených úpravách, případně při detekci a opravě natočení je dalším krokem extrakce samotných kontur a linek. Ta je prováděna za pomoci tzv. *blob detection* - jedná se v podstatě o nalezení kontur či hlouběji nepopsaných objektů v dokumentu. Po nalezení takových objektů v dokumentu je například na základě průměrné výšky kontur v regionu odhadnuta výška znaků a kontury jsou poté filtrovány dle velikosti tak, aby se zamezilo zkreslování dalších výsledků nevalidními výskyty (šum, poškození, interpunkce přesahující



(a) Binarizace [25, převzato]

(b) Natočení [25, převzato]

Obrázek 2.1: Preprocessing

mezi linkami). Na základě takto filtrovaných kontur a jejich seřazení dle x-ových a y-nových souřadnic je možno detekovat paralelní a nepřekrývající se linky. Jakmile jsou jednotlivé kontury přiřazeny linkám, je možno na základě mediánů rozměrů jednotlivých kontur vytvořit vodicí linky. Tyto linky pak slouží ke zpětnému umístění dříve odfiltrovaných malých kontur [30].

2.1.2 Klasifikační vlastnosti

Extrakce klasifikačních vlastností neboli *feature extraction* je obecný pojem z disciplíny *pattern matching/pattern extraction*, jehož cílem je zkoumání a nalezení takových vlastností, které jsou co nejvhodnější pro nalezení správného vzoru. Tento pojem se však určitě neváže jen na OCR, ale jak bylo zmíněno výše, jedná se o obecný pojem/disciplínu, která má širokou škálu využití. V obecném slova smyslu není OCR nic jiného než zpracování obrazu. Valná vetšina používaných metod pro OCR je tedy úzce spjata právě s obecným zpracováváním obrazu. Tato část práce je věnována přehledu některých obecných vlastností, které mohou být použity pro pozdější rozpoznání konkrétního znaku systémy OCR. Popisuje základní myšlenku jednotlivých vlastností a případně možnosti jejich použití pro rozpoznávání znaků. Na rozdíl od *template matching* se tyto vlastnosti snaží v co nejvyšší možné míře ignorovat variabilitu jednotlivých znaků či variabilitu jednotlivých dokumentů. Obecně se uvádí invariance vůči různým druhům transformací - tedy invariance vůči rotaci, invariance vůči změně velikosti, invariance vůči posunutí atd. Čím větší míru invariance vlastnost má, tím je obecnější a tím více zanedbává obrazové transformace a je vetší šance na rozpoznání hledaného znaku, či obecně hledané vlastnosti.

Momenty

Jednou z typických vlastností sloužících pro popis obrazu a pro popis objektu v rámci zpracování obrazu, jeho rozpoznání a klasifikaci jsou momenty či momentové invarianty (funkce). Nejjednodušeji představitelným příkladem momentu může být například průměr barev 8-okolí daného pixelu. Toto 8-okolí pixelu je reprezentováno nějakou číselnou hodnotou generovanou na základě histogramu. Toto číslo je pak porovnáno s jinými hodnotami generovanými na základě dalších histogramů odpovídajících porovnávaným 8-okolím. Tyto

generované hodnoty jsou pak porovnávány a je hledána co nejlepší shoda. Obecně je invariantní rozpoznávaní objektu realizováno klasifikací v multidimensionálním momentově invariantním prostoru vlastností.

Ming-Kuei Hu v roce 1962 představil celkem 7 2-dimenzionálních funkcí popisujících tvar objektu v obraze vycházejících z geometrických a centrálních momentů. Tyto funkce, stejně tak jako centrální momenty, jsou invariantní vůči posunutí, změně měřítka, rotaci. Invariance je dosaženo normalizací momentu vzhledem k těžišti objektu a velikosti. Geometrický neboli regulární moment je nejpoužívanějším typem momentu a je definován jako rovnice 2.1.

$$m_{pq} = \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} x^p y^q f(x,y) dx dy \tag{2.1}$$

kde p, q jsou přirozená čísla, $(p+q)$ je řád momentu, x, y jsou souřadnice bodu v obrazu a $x^p y^q$ je bázová funkce. Z něj pak částečnou normalizací dostáváme centrální moment definován rovnicí 2.2.

$$m_{pq} = \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} (x-\bar{x})^p (y-\bar{y})^q f(x,y) d(x-\bar{x}) d(y-\bar{y}) \tag{2.2}$$

Dále se centrální moment normalizuje vůči změně velikosti a lze z něj odvodit 7 nelineárních funkcí sloužících k popisu vlastností objektu v obraze. Z toho lze odvodit například podobnost dvou objektů, či obrazových vstupů, tedy i podobnost pro porovnávání segmentů znaků v rámci OCR. Dalšími typy momentů jsou například *Zernike Moments, Legendre Moments, Tchebichef Moments* [12, 16, 20].

Směrové vlastnosti

Jednou z dalších obecných vlastností vycházejících z obecné disciplíny zpracovávání obrazu je histogram orientací. Ten je velmi snadno využitelný také pro rozpoznávání znaků. Histogram orientací je vypočítáván pro sub-oblasti znaku. Oblast znaku je tak například rozdělena do matice zón o velikosti 4x4 a pro každou zónu je vypočítán úhel kontury zkoumaného znaku. Tento úhel je pak zařazen do kategorie úhlů definované dle rozpětí a počet zón spadajících do každé kategorie úhlů je poté považován za vlastnost sloužící k rozpoznání znaku. Příslušnost do kategorie pak určuje váhu v daném rozměru vektoru popisujícím daný znak (histogram orientací při OCR na obr. 2.2).

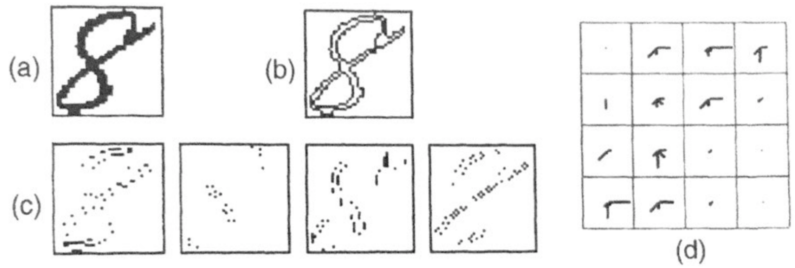


FIGURE 3.1 (a) Character image; (b) contour of character; (c) orientation planes; (d) local orientation histograms.

Obrázek 2.2: Histogram orientací při OCR [12, převzato]

Houghova transformace

Jedná se o robustní statistickou metodu pro detekci přímek, křivek a obecně kuželoseček z obrazových podkladů. Cílem této metody je nalézt a kategorizovat objekt (konturu) do dané třídy (přímka, elipsa, či jiná kuželosečka), případně vyhladit nedokonalosti a nespojitosti ve křivkách způsobených například předzpracováním obrazu. Jako nejjednodušší příklad je často úváděna detekce přímky v obraze. Při standardní Houghově transformaci jsou všechny body obrazu namapovány do parametrického prostoru a tedy při detekci přímky budeme namísto směrnicového vyjádření přímky v obrazovém prostoru (rovnice 2.3)

$$y = mx + b \tag{2.3}$$

používat normálový tvar v parametrickém prostoru určený rovnicí 2.4

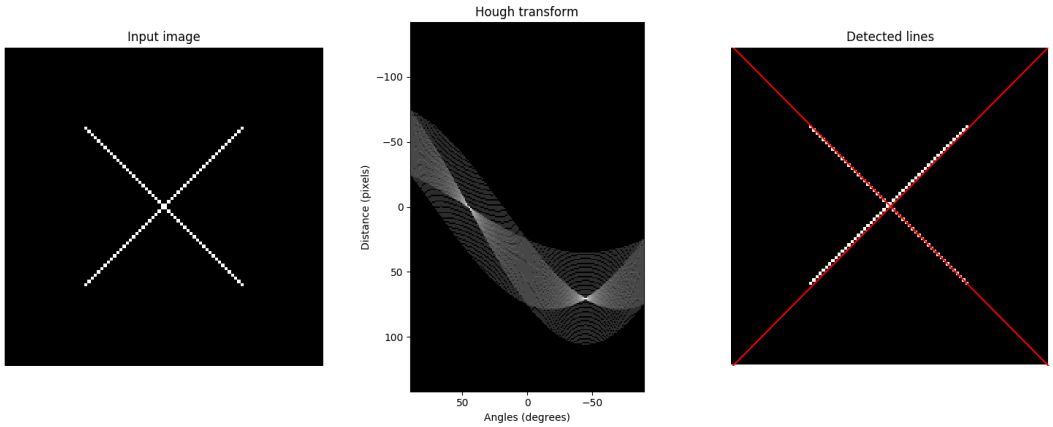
$$\rho = x\cos\theta + y\sin\theta \tag{2.4}$$

kde ρ je délka normály z počátku soustavy souřadnic k přímce, θ je úhel této normály vůči ose x a x, y jsou souřadnice bodu. Úsečka reprezentována v obrazovém prostoru například směrnicovou rovnicí zmíněnou výše je tedy namapována na jeden jediný bod v prostoru parametrickém, a sice $(\theta_{max}, \rho_{max})$. Při detekci jsou však (θ, ρ) z normálového vyjádření přímky neznámými veličinami a jsou známé pouze souřadnice zpracovávaných bodů x, y. Při dosazení obrazového bodu kontury do rovnice 2.4 dojde v parametrickém prostoru k vytvoření spojité křivky (sinusoidy) reprezentující všechna možná řešení (θ, ρ). Při zpracování více obrazových bodů z kontury dojde v parametrickém prostoru k protnutí výsledných spojitých křivek v jediném bodě. Tento bod nese informace o $(\theta_{max}, \rho_{max})$. Po dosazení těchto hodnot do rovnice 2.4 je možné získat přesný popis kontury ve vstupním obrazu.

Reálně se však samozřejmě nemohou zpracovávat pouze obrazové body kontury, která zatím není známa, jelikož cílem je její nalezení pomocí Houghovy transformace. Je výhodné předzpracovat obraz detektorem hran pro extrakci možných kontur. K dispozici je tedy dvouhodnotový obraz zachycující jednotlivé hrany. Při samotné detekci se zpracovávají všechny obrazové body a pro každý pixel zachycující detekovanou hranu je vypočteno ρ, kde za θ jsou dosazeny hodnoty v dohodnutém rozmezí s dohodnutým krokem. Poté se pro každý bod v parametrickém prostoru (θ, ρ) inkrementuje hodnota, a tím je získán tzv. Houghův prostor. V tomto prostoru se nacházejí maxima, která reprezentují průsečíky spojitých křivek popsaných výše. Z těchto průsečíků jsou pak odečteny hodnoty $(\theta_{max}, \rho_{max})$ a výsledné přímky jsou vykresleny zpět do vstupního obrazu [15, 12, 18]. Příklad procesu zpracování je na obr. 2.3. Houghovou transformací tedy lze detekovat kuželosečky a přímky v obraze, což se hodí v první fázi OCR při předzpracování dokumentu. Dále jejich tvar a počet může být i markantem pro rozpoznání znaku za předpokladu, že oblast jednoho znaku je již detekována při výběru zájmových oblastí.

Aproximace tvaru

V tomto případě se jedná o převod rastrového tvaru objektu do formy vektorové, tedy v případě OCR do 2-rozměrného polygonu. Přestože nejpřesnější aproximace tvaru objektu v rastrovém obraze by byla získána přiřazením každému pixelu tvaru jeden segment výsledného polygonu, cílem polygonální aproximace je však počet segmentů snížit na minimum s ohledem na kvalitu aproximace. Jednou z technik je aproximace tvaru pomocí krátkých úseček, které jsou vzájemně propojeny do jednoho celku, přičemž každá z daných úseček má definovaný úhel natočení tak, aby byla aproximace co nejvěrnější originálnímu tvaru - tzv. *chaining*.



Obrázek 2.3: Houghův prostor [29, převzato]

Další významnou aproximační technikou jsou Bézierovy křivky. Bézierova křivka je obecně založena na množině kontrolních bodů, které definují zakřivení segmentů křivky. Křivka těmito body přímo neprochází (kromě prvního a posledního bodu). Mějme tedy dva body definující Bézierovu křivku P_0 a P_1 - zakřivení křivky v bodě P_0 je stejné jako zakřivení křivky spojující tyto dva definované body. Bod P_1 tedy svou relativní pozicí vůči bodu P_0 určuje tvar křivky. Nejpoužívanějším příkladem Bézierovy křivky je křivka třetího řádu. Ta je určena 4 body, kde druhý a třetí bod určují směr tečny ke křivce v krajních bodech, kterými křivka prochází. Obecně je křivka definována jako váhový součet jednotlivých bodů, kde jednotlivé váhy bodů jsou Bernsteinovy polynomy definované rovnicí (2.5) jako:

$$B_i^n(t) = \binom{n}{i}(1-t)^{n-i}t^i \tag{2.5}$$

z toho pak obecná rovnice (2.6) Bézierovy křivky:

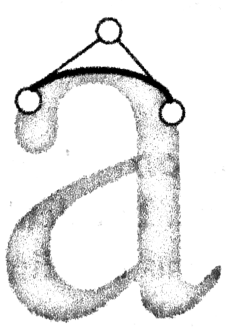
$$\bar{P}(t) = \sum_{i=0}^{n} P_i B_i^n(t), 0 \leq t \leq 1 \tag{2.6}$$

kde n je řád křivky, $i = 0, 1, ..., n$, $B_i^n(t)$ je Bernsteinův polynom (uveden výše) a P_i je definovaný bod křivky [12]. Bézierovy křivky se dají použít pro popis tvaru jednotlivých znaků, příklad je uveden na obr. 2.4.

Další klasifikační vlastnosti

Mezi další klasifikační vlastnosti se řadí například Fourierovy deskriptory, které popisují tvar obrazového objektu pomocí frekvencí. Dle literatury jsou používané pro svou odolnost vůči šumu a přímočarou normalizaci. Fourierovy deskriptory využívají 1-rozměrnou hraniční reprezentaci tvaru. Ta obsahuje různé popisy daného tvaru, mezi nimi například komplexní souřadnice - převádí klasické souřadnice (x, y) do tvaru $z(t) = [x(t) - x_c] + j[y(t) + y_c]$, vzdálenost geometrického středu, popis zakřivení a další.

Za klasifikační vlastnost se dá také do jisté míry považovat histogram kontury a její okolí v obraze. Histogram vyjadřuje zastoupení jednotlivých barevných hodnot v obraze. Na základě histogramu se tedy dá odhadnout, že objekt na obrazovém podkladu s vyšším počtem typicky tmavých pixelů bude spíše znak *B, G, M* nežli znak *I* či *C*.



Obrázek 2.4: Bézierova křivka - definice znaku [12, převzato]

Existuje samozřejmě mnoho dalších vlastností, ze kterých lze vybrat ty, které se pro daný úkol nejvíce hodí, či transformační metody, které zlepšují úspěšnost zde popsaných extraktorů vlastností. Jedná se například o různé lineární, afinní, projektivní, perspektivní transformace obrazu, *Thinning* (ztenčování linek), nebo snižování složitosti popsaných metod [12]. Výběr relevantních vlastností používaných pro detekci samotného znaku jsou popsány v následující části práce.

2.1.3 Selekce klasifikačních vlastností

Vlastností popisujících daný vzor či znak může být velké množství. Je evidentní, že některé z vlastností jsou více vhodné pro danou klasifikační úlohu nežli jiné. Množinu všech vlastností je možné si představit jako n-dimenzionální prostor, kde každý rozměr představuje jednu vlastnost. Těch mohou být stovky až tisíce. Práce v takovém prostoru je však výpočetně náročná (výpočetní náročnost a nutný počet učicích vzorků roste exponencionálně vůči počtu dimenzí), a proto se na základě daných kritérií dimenzionalita prostoru redukuje a v ideálním případě vyřazuje méně vhodné, či výpočetně náročné vlastnosti - probíhá jejich selekce. Existuje několik přístupů k selekci vlastností.

Prvním z přístupů je filtrování. V tomto případě je množina zkoumaných vlastností redukována před samotným během kasifikátoru. Klasifikační vlastnosti jsou vybírány na základě vnitřních parametrů, jež by měly být na základě statistického předpokladu významově co nejblíže klasifikační úloze. Výhodou je nezávislost filtru na samotném klasifikátoru.

Dalším přístupem je obalení sub-množiny vlastností učicím algoritmem. Výběr klasifikačních vlastností pak probíhá na základě účinnosti učícího algoritmu. Čím je učící algoritmus účinější, tím je vybraná sub-množina klasifikačních vlastností lepší pro daný klasifikační úkol.

Hybridní selektory se snaží potlačit nevýhody výše zmíněných přístupů k selekci a naopak vybrat to nejlepší z nich. Jedná se například o více-průchodové selektory - vyhodnocení výsledků učícího algoritmu na vybrané submnožině, poté filtrování. Dalším hybridním přístupem je tzv. *ReliefF-GA-Wrapper* zahrnující také například využití genetického algoritmu pro ohodnocování.

Na základě vybraného přístupu selekce je vyhodnocení kvality sub-množiny (ne)závislé na daném klasifikátoru. Kvalita sub-množiny je evaluována na základě několika kritérií, například inter a intra třídní variability, pravděpodobnostní vzdálenosti, entropie a další. Obecně jsou přístupy využívající filtrování před spuštěním samotného klasifikátoru považovány za výhodnější, jelikož nezávisí na klasifikátoru a jsou výpočetně méně náročné [12].

2.1.4 Zpracování vektorů vlastností - rozpoznání znaků

Po vybrání zkoumaných vlastností a jejich získání z daného vzorku je nutné je pomocí klasifikátoru kategorizovat a získat pravděpodobnou hodnotu znaku. Mějme tedy extrahované vlastnosti namapovány do n-rozměrného prostoru vlastností. Cílem a výstupem klasifikace je každému takovému bodu přiřadit tzv. štítek nebo skóre vyjadřující příslušnost jednotlivým kategoriím či třídám. Používané klasifikátory je dle přístupu možné kategorizovat do několika generických skupin:

- statistické metody
- metody využívající neuronové sítě
- *support vector machines*
- strukturální klasifikátory
- kombinované klasifikátory

přičemž ty se liší ve zpracování a vyhodnocování jednotlivých vlastností, či v přístupu k vylepšování přesnosti klasifikátoru [12].

Z popsaných přístupů a technik v této kapitole je evidentní, že OCR je netriviální mezioborová úloha využívající široké spektrum vědních disciplín. Od zpracování obrazu a grafiky, přes matematickou analýzu a práci v multidimensionálních prostorech, až po neuronové sítě. Dosavadní popis v této kapitole je tedy pouze obecným přehledem velmi rozsáhlé problematiky, která se vyvíjela dlouhá léta. Z pohledu této práce slouží však pouze pro pochopení základních principů a taky jako nástin technik, které mají široký přesah a jsou využívány i níže popisovanými metodami.

2.2 Praktické využití OCR

Z výše popsaného je evidentní, že implementace vlastního systému OCR by byla velmi obtížná úloha pravděpodobně nad rámec náročnosti této práce. Naštěstí existují dostupné systémy, které OCR implementují a jsou veřejně dostupné a lehce použitelné. Jsou jimi například *Tesseract* a *OCRopus*, nebo zpoplatněný *ABBYY*. Konkrétněji je v praktické části této práce využit systém *Tesseract*. Volba na tento konkrétní systém padla ze 3 důvodů:

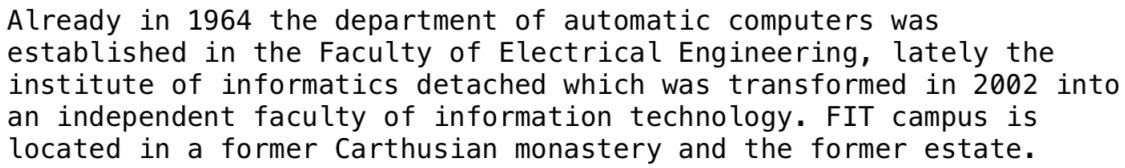
- Jedná se o open source a na rozdíl od *ABBYY* je zdarma
- *OCRopus* původně využíval *Tesseract* jako své jádro a jednalo se pouze o nadstavbu
- Předchozí kladná zkušenost se systémem *Tesseract*

Vstupem systému *Tesseract* je obraz ve vysoké kvalitě (v této práci typicky převedeno pdf do 300 dpi), výstupem je rozpoznaný text. Volitelně je také možno získat například informace o pozicích jednotlivých slov apod. V této práci však v rámci předzpracování stačí získat zmíněné textové informace a ty mohou být dále analyzovány.

Výhodou systému *Tesseract* bezesporu je, že sám o sobě implementuje a provádí všechny potřebné úkony k rozpoznávání textu, čili i výše zmíněné předzpracování, jako například převod do černobílé, potlačení rotace apod. Sám také získá potřebné klasifikační vlastnosti a zpracovává i jejich vektory. Drobnou nevýhodou je naopak doba běhu. Typická strana A4 trvá na jednoprocesorovém PC cca 12 sekund. Toto se podařilo v této práci alespoň

částečně optimalizovat tak, že je Tesseract spouštěn pouze na stránky, ze kterých nebylo možné získat textová data jiným způsobem. Toto bude dopodrobna vysvětleno v kapitole 5.

Úspěšnost systému OCR závisí především na kvalitě vstupních dat. Pokud máme textový soubor převeden počítačově do obrazové podoby, je úspěšnost samozřejmě daleko vyšší než u špatně naskenovaného dokumentu. Níže (obr. 2.5 a obr. 2.6) jsou uvedeny dva příklady získaných dat pomocí *Tesseract*.

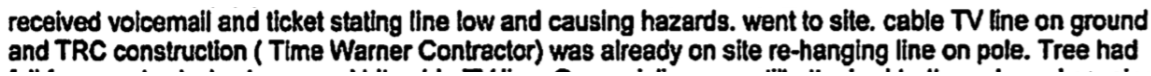


Obrázek 2.5: Kvalitní vstup pro OCR [vlastní]

Výsledkem analýzy pomocí *Tesseract* takového souboru je:

> Already in 1964 the department of automatic computers was established in the Faculty of Electrical Engineering, lately the institute of informatics detached which was transformed in 2002 into an independent faculty of information technology. FIT campus is located in a former Carthusian monastery and the former estate.

Naproti tomu při nekvalitním vstupu se úspěšnost lehce zhorší.



Obrázek 2.6: Nekvalitní vstup pro OCR [vlastní]

Výsledkem analýzy pomocí *Tesseract* takového souboru pak je:

> received voloemall and ticket stating line low and causing hazards. went to site. cable TV fine on ground 9quch capstrycttont Time Werger gamer) gas alteagy on site ?9419'an l'lne'on pele. Tt_ee hag

Jednoduše si lze představit, že vstupní obraz může mít ještě mnohonásobně horší kvalitu a rozpoznání textu dopadne mnohem hůře. V takovém případě je samozřejmě pro zjištění podobnosti podčástí dokumentů k jejich rozdělování vhodné brát v potaz také jiné markanty než pouze textovou reprezentaci dokumentu. Například lze porovnávat obrazová podpobnost. O tomto však více v kapitole 4. Nicméně kvalita vstupního obrazu je zásadní pro úspěšnou analýzu OCR, potažmo také pro úspěšnou analýzu a automatizované rozdělování souborů implementované touto prací.

V této kapitole bylo popsáno získávání nestrukturovaných textových dat z dokumentů, které obsahují texty reprezentované obrazovou formou. Primárně bylo popsáno OCR, jeho jednotlivé kroky, bylo uvedeno několik ukázek vlastností pro klasifikaci znaků a byly popsány metody používané k samotnému rozpoznávání. Byly uvedeny příklady existujících systémů pro OCR, stejně tak jako byla předvedena úspěšnost systému Tesseract, který je použit v implementační části této práce.

Kapitola 3

Extrakce relevantních informací z textových dat

Tato kapitola se zabývá získáváním relevantních informací z dokumentů obsahujících textová data. Zkoumané dokumenty mohou buď obsahovat textová data samy o sobě, nebo mohla být získána pomocí OCR, které je popsáno v předchozí kapitole. Zpracováním takovýchto nestrukturovaných dat se zabývá obor zpracování přirozeného jazyka (*Natural Language Processing*) - NLP. Opět se jedná o velmi široký vědní obor, ze kterého je v této práci popsána pouze velmi malá část potřebná k úspěšné realizaci zadaného systému. Z pohledu této práce je tedy nejzajímavější získávání strukturovaných informací, které lze z textových dokumentů získat pro lepší kategorizaci a rychlejší zpracování. Základními skupinami popisných informací jsou například [9]:

- klíčová slova
- zainteresované osoby
- datumy
- lokace
- sentiment
- ...

3.1 Extrakce klíčových slov

Klíčová slova jsou termy, které v co největší míře zachycují obsah dokumentu, jeho téma a zaměření. Slouží především pro rychlou orientaci mezi vícero dokumenty, jejich vyhledávání, kategorizaci apod. Současně však musí být pro efektivní práci počet klíčových slov o několik řádů menší, než je délka zkoumaného dokumentu. O to důkladněji je potřeba klíčová slova pro dané dokumenty vybírat/extrahovat.

Existuje samozřejmě několik metod pro extrakci klíčových slov z textů, v zásadě se však dělí na 2 skupiny. Jsou jimi metody založené na *corpus* (slovníku) a metody nezávislé na stavu a obsahu slovníku - tzv. *document-oriented*. Ty předpokládají neustálé změny ve slovníku (například přidáváním vědeckých článků do databáze slovník v čase roste) a generují neustále stejná klíčová slova [11]. Jedním ze široce používaných algoritmů je *RAKE - Rapid Automatic Keyword Extraction*.

3.1.1 RAKE - Rapid Automatic Keyword Extraction

RAKE je jednoprůchodový algoritmus nezávislý na jazyku či informační doméně dokumentu a vychází z předpokladu, že klíčové slovo či spíše fráze často obsahuje více slov, ale jen minimální počet tzv. *stop words* s minimálním sémantickým významem - v angličtině jsou to např.: *and, the, of*. Jedním ze vstupů algoritmu je pak seznam takových slov a dalších oddělovačů (např. interpunkce). Tato slova jsou pak použita k rozdělení vstupního textu, jsou vynechána a rozdělené podčásti slouží jako kandidáti na klíčová slova:

Vstup: . . . solutions and algorithms. . . These criteria and the corresponding algorithms Výstup: [solutions, algorithms, criteria, corresponding algorithms]

Následně je sestavena tabulka (tab. 3.1) společných výskytů slov ve vygenerovaných frázích (klíčových slovech).

-	algorithms	corresponding	criteria	solutions
algorithms	2	1	-	-
corresponding	1	1	-	-
criteria	-	-	1	-
solutions	-	-	-	1

Tabulka 3.1: Společné výskyty slov

Na základě takové tabulky je možno vypočítat stupeň (*deg*), frekvenci (*freq*) a výsledné skóre (*score*) pro každé slovo (w). Frekvence značí, kolikrát se samotné slovo v textu vyskytlo (čísla na diagonále), preferuje tedy slova na základě počtu jejich výskytu. Stupeň započítává i výskyty v delších frázích (suma řádků) a preferuje slova vyskytující se v delších frázích. Výpočet konečného skóre je pak stanoven rovnicí 3.1

$$score(w) = \frac{deg(w)}{freq(w)} \tag{3.1}$$

a bere v potaz obě veličiny. Poté jsou slova seřazena dle generovaného skóre a je vybráno x nejlépe ohodnocených klíčových slov ze všech slov v tabulce společných výskytů, například jako $x = \frac{word\_count}{3}$ [11].

3.2 Rozpoznávání pojmenovaných entit

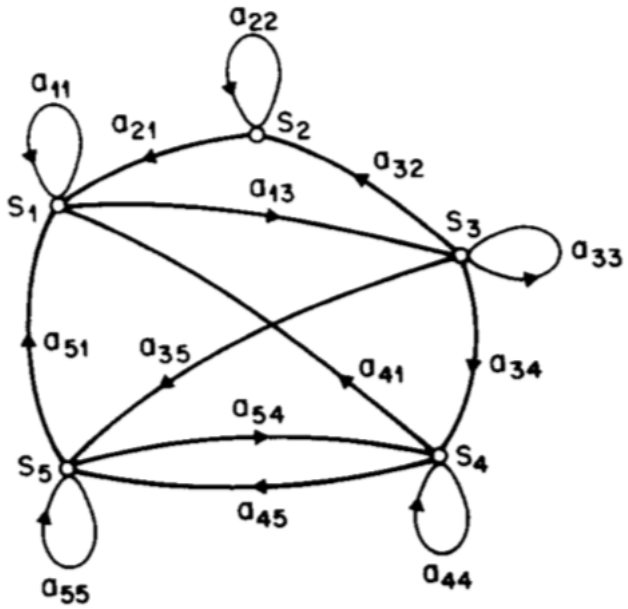
Přestože jsou klíčová slova celkem dobrou a používanou metrikou, jsou považována za pouze velmi základní způsob analýzy dokumentu. Pokročilejším přístupem je pak tzv. rozpoznávání pojmenovaných entit neboli *Named Entity Recognition - NER*. To je jedním z fundamentálních způsobů získávání informací z textových dokumentů. Na rozdíl od klíčových slov generovaných bez sémantického významu se NER snaží jednak získat klíčové entity, ale také jim přiřadit sémantický typ. Typicky uváděnými typy entit jsou osoba (*PER*), organizace (*ORG*), lokace (*LOC*) a ostatní (*MISC*). Dalším možným krokem je zjištění vztahů mezi takovými entitami. Poté lze tedy ze zadaného textu vyjádřit *kdo, kdy, jak*.

Existuje několik typů analýzy - a sice analýza založena na jednoduchém slovníkovém vyhledávání, *rule based* analýza pužívající definovaná pravidla, jež jsou zpracovávána pomocí konečného automatu - fungujících podobně jako regulární výraz a modernější analýza využívající strojové učení. Nevýhodou prvních dvou přístupů je závislost na vstupním textu a nízká míra přenositelnosti. Při výskytu nového vstupního dokumentu množiny je nutné

upravit existující slovník, či pravidla tak, aby bylo docíleno maximálního výkonu daného systému [32, 28]. Velkou výhodou přístupů využívajících strojové učení je samozřejmě jejich udržovatelnost. Typickými a pravděpodobně nejlepšími zástupci tohoto přístupu jsou systémy využívající skryté Markovovy modely (*Hidden Markov Models - HMM*) [32].

3.2.1 Markovovy modely

Markovův model je systém, který je určen N stavy S. V pevně daných diskrétních časových intervalech dochází v takovém systému ke změně (akci A) stavu, kde nový stav může být totožný s původním na základě množiny pravděpodobností daných ke každému stavu a možnému přechodu. Pokud budeme uvažovat Markovův proces 1. řádu, následující stav závisí pouze na aktuálním přechodu a stochastickém výběru přechodu z tohoto stavu. Jedná se tedy v podstatě o pravděpodobnostní model konečného automatu (obr. 3.1). Často je také možné se setkat s pojmem Markovův rozhodovací proces - tento musí splňovat Markovovu vlastnost o nezávislosti stavů popsanou výše [4].



Obrázek 3.1: Markovův model [26, převzato]

Konkrétněji se u NER jedná o typ stochastického Markovova modelu/procesu, kdy jsou jeho stavy skryty pozorovateli a viditelné jsou pouze výstupy závisející právě na jeho stavech - také nazýván jako *Hidden Markov Model*, nebo *Partially observable Markov Model*. Tyto pozorovatelné výstupy jsou pravděpodobnostní funkcí daného stavu a je tedy možné na základě jejich sekvence odhadnout posloupnost vnitřních skrytých stavů [26, 32]. Právě tento odhad stavů je zásadní pro rozpoznávání entit, kde se minimálně v první fázi - *chunking* využívá právě pravděpodobnostní distribuce stavů. Dalšími možnými přístupy jsou *KRDL's, Maximum Entropy Markov Model, Decision Tree* [32].

3.2.2 Využití HMM v obecném systému rozpoznávání entit

Obecně se jakýkoliv systém hledající entity se sémantickým významem skládá ze 4 částí. Jsou jimi [28]:

- *chunking* - rozdělování vstupního textu
- *inference* - nalezení a případná kategorizace entity
- *non-local dependencies* - identifikace vícero shodných entit v jednom nebo více dokumentech s různou binární podobou
- *external resources* - užití slovníkových záznamů jako klasifikačních vlastností

Chunking a Inference

Na rozdíl od hledání klíčových slov, nelze vstupní text rozdělit na základě předdefinovaných statických oddělovačů nebo reprezentovat entity jako jednoslovné. Stejně tak nelze jednoduše vyfiltrovat tzv. *stopwords*. Typicky víceslovné jsou například názvy organizací, jako *Brno University of Technology* - tento příklad navíc obsahuje i *stopword* of, které by bylo při získávání klíčových slov pravděpodobně vyfiltrováno jakožto slovo s minimálním sémantickým významem. Uvedený řetězec (a jemu podobné) musí být rozpoznán jako jediná nedělitelná entita nesoucí právě jméno organizace. K rozdělování vstupního textu (chunking) a nalezení entit (inference) se tedy používají jiné metody, například zmíněné *HMM*. Ty jsou mimo jiné základem pro tzv. *Part-of-speech (POS) tagging* - tedy pro identifikaci slovních druhů.

A právě distribuce POS je jedním ze stavebních kamenů pro rozdělování vstupního textu, a tedy i pro rozpoznávání entit. Cílem je samozřejmě najít co nejpřesnější distribuci rozdělujících značek ve vstupním textu a identifikovat typy entit s co největší přesností. K tomu se využívá řada algoritmů jako *greedy left-to-right decoding*, *beamserach* a *Viterbi* [28]. Poslední zmíněný algoritmus (*Viterbi*) je považován za nejefektivnější a je používán právě k nalezení nejpravděpodobnější distribuce rozdělujících značek vstupního textu ve stavovém prostoru všech možných distribucí rozdělujicích značek založených na pravděpodobnostech přechodů *HMM* - někdy nazýváno také jako Viterbiho cesta. To navíc zvládne s lineární časovou složitostí a nalezne optimální sekvenci rozdělujicích značek [23].

Non-local dependencies

Důležitým prvkem inference popsané výše je také stálost přiřazení typů entit v rámci dokumentu - tedy stejné řetězce by měly být identifikovány vzhledem ke kontextu, ve kterém se nacházejí, shodně. Důležitá je také shodná identifikace entit, které mohou být reprezentovány jinými řetězci, tedy například identifikovat, zda se jedná ve všech případech vzhledem k uvedenému kontextu o osobu (identifikace na obr. 3.2): $BLINKER \rightarrow PER$, *Reggie Blinker* $\rightarrow PER$ a *Blinker* $\rightarrow PER$ [28]. V literatuře se objevuje několik vlastností k řešení popsaného problému, ty jsou zevrubně popsány níže.

Context aggregation vytváří pro každou rozeznanou entitu kontext, ve které se nachází. Jednoduše se snaží popsat výskyt entity podobně jako *„the token appears before a company marker such as ltd.“* [28]. Takový kontext může být sestrojen automatizovaně pomocí klouzavého okénka a sloužit jako agregace dílčích kontextů do výsledné popisující vlastnosti.

Two-stage prediction aggregation využívá toho, že některé entity se vyskytují v jednoduše identifikovatelných kontextech. Tyto entity lze pak použít pro predikci ostatních entit. Nejprve tedy proběhne základní NER analýza a výsledné kontexty jsou použity pro analýzu druhé úrovně.

Extended prediction history se od předchozích dvou technik liší způsobem zpracování jednotlivých entit - a sice uvádí, že v úvodních částech textových dokumentů dochází k vyššímu

SOCCER - [PER BLINKER] BAN LIFTED .
[LOC LONDON] 1996-12-06 [MISC Dutch] forward [PER Reggie Blinker] had his indefinite suspension lifted by [ORG FIFA] on Friday and was set to make his [ORG Sheffield Wednesday] comeback against [ORG Liverpool] on Saturday . [PER Blinker] missed his club's last two games after [ORG FIFA] slapped a worldwide ban on him for appearing to sign contracts for both [ORG Wednesday] and [ORG Udinese] while he was playing for [ORG Feyenoord].

Obrázek 3.2: Problém konzistence přiřazení typu entitám [28, převzato]

počtu nalezení shod než v dalších částech. Údajně je to způsobeno tím, že čím blíže je osoba, lokace atd. začátku textu, tím větší je pravděpodobnost použití kanonických pojmenování, zatímco později v textu mohou být odkazovány zájmeny a podobně. Z toho lze usuzovat, že pro odhad typu entity v textu se dají použít její předchozí výskyty. Tedy pokud je zkoumána entita x_i, současně bylo klasifikováno $x_1, \ldots, x_{i-1}$ předchozích souhlasných entit, na základě odhadů $y_1, \ldots, y_{i-1}$ je použita distribuce dříve přiřazených typů. Tedy pokud se v přechozích 5 predikcích třikrát vyskytoval typ *LOC* a dvakrát typ *ORG*, jednoduše je vzata pravděpodobnost $\frac{2}{5}$ versus $\frac{3}{5}$. Tento přístup však nelze použít současně s Viterbiho algoritmem pro rozdělování vstupního textu, jelikož je nutné aby text byl zpracováván sekvenčně zleva doprava.

Co se týká výkonu, popsané vlastnosti dosahují podobných výsledků při různých testovacích souborech [28].

External resources

Jak bylo zmíněno v úvodu této kapitoly, existují také přístupy využívající pouze slovníkové vyhledávání. Ty dosahují celkem dobré úspěšnosti - kolem 70 %. V kombinaci se strojovým učením a popsanými metodami výše je však možné dosáhnout úspěšnosti až kolem 90 %. Současně je tedy i jednoduchých slovníkových záznamů využívano pro identifikaci typu entity jako jedné z vlastností klasifikace. Typicky je využíváno více různých slovníků, z nichž každý obsahuje miliony záznamů. Není neobvyklé použití dat z webu, jako například online encyklopedie *Wikipedia*. Srovnání úspěšnosti jednoho ze systému NER při použití různých klasifikačních vlastností je v tabulce na obrázku 3.3. CoNLL03 v tabulce odkazuje na *Conference on Computational Natural Language Learning (CoNLL) "shared task"* z roku 2003, kde cílem bylo právě NER a stalo se standardem pro měření výkonosti a kvality NER systémů. Podobným srovnávacím příkladem je i MUC7 [28].

3.3 Praktické využití klíčových slov a NER

V rámci implementační části této práce byl využit již existující *engine* pro získávání zmíněných relevantních dat využívající popsané techniky. Na výběr je hned z několika možností

	Component	CoNLL03 Test data	CoNLL03 Dev data	MUC7 Dev	MUC7 Test	Web pages
1)	Baseline	83.65	89.25	74.72	71.28	71.41
2)	(1) + External Knowledge	88.55	92.49	84.50	83.23	74.44
3)	(1) + Non-local	86.53	90.69	81.41	73.61	71.21
4)	**All Features**	**90.57**	**93.50**	**89.19**	**86.15**	**74.53**
5)	All Features (train with dev)	90.80	N/A	89.19	86.15	74.33

Obrázek 3.3: Úspěšnost NER při využití různých klasifikačních vlastností [28, převzato]

od *Aylien API* přes *NLP Compromise* až po *Google Cloud Natural Language API*. Většina z nich nabízí srovnatelné možnosti jako analýzu sentimentu, získání klíčových slov, získání pojmenovaných entit atd. Z pohledu této práce se však zdál jako nejvhodnější kandidát *Aylien API*, jež je používán právě pro získávání relevantních informací z již získaných podčástí automatizovaně, či manuálně rozděleného dokumentu. Především jsou získávány lokace a osoby vyskytující se v daných dokumentech. Ty jsou pak vynášeny na časovou osu pro vizuálně jednoduchou analýzu dokumentů z uživatelského pohledu.

V této části práce byly popsány techniky pro získávání strukturovaných dat z textových dokumentů, zejména pak extrakce klíčových slov a extrakce pojmenovaných entit z textů. U extrakce klíčových slov byl vybrán a popsán jeden zástupce zmiňovaný v odborné literatuře jako jeden z nejlepších. Etrakce pojmenovaných entit byla rozebrána poněkud podrobněji. Byly popsány jednotlivé fáze získávání entit a přidružené algoritmy, které se dají pro každou fázi použít. Dále bylo popsáno využití těchto metod v implementovaném systému v rámci této práce.

Kapitola 4

Rozdělování dokumentů

Jedním z hlavních cílů této práce je využití nejen výše získaných informací k automatizovanému rozdělování vícero dokumentů spojených dohromady a jejich kategorizace. Vícero dokumentů mohlo být spojeno například skenováním, sloučením vícero elektronických pdf dokumentů, tiskem a podobně. Takové rozdělování dokumentů vlastně není nic jiného než vyhodnocování podobnosti jednotlivých podčástí (stránek) dokumentu. Tato podobnost může být vyhodnocována na základě statistické analýzy výskytů slov v textu, nebo sémantické analýzy podčástí dokumentu, pokud jsou známa sémantická (textová a strukturovaná) data. Je však velmi málo pravděpodobné, že je možné získat sémanticky anotovaná data ze všech podčástí daného dokumentu - některé části mohou obsahovat např. pouze obrázky. Proto se nabízí, aby dalším markantem byla podobnost grafických komponent, rozložení textu na stránce, barevná reprezentace atd. V obecném kontextu se právě odhadu podobnosti dokumentů mimo jiné věnuje tzv. *Document Clustering*. Ten sice zkoumá podobnost vícero dokumentů, pro případ rozdělování se však jím využívané techniky dají zužitkovat pro rozdělování dokumentů na podobné části. Jednoduše stačí reprezentovat podčásti nebo jednotlivé stránky zkoumaného dokumentu jako samotné dokumenty a nehledat jejich podobnost, ale jejich rozdílnost.

4.1 Document clustering

Hlavním cílem disciplíny *Document clustering* (shlukování dokumentů) je organizování dokumentů do skupin (například) dle jejich tématu pro lepší vyhledávání a organizaci, sumarizace a extrahování témat dokumentů. Jako jedno z možných využití bývá také uváděna organizace výsledků vyhledávání na internetu. Podobnost dokumentů může být určena například vzhledem ke slovníku a počtu výskytů jednotlivých slov v dokumentech [8], nebo na základě sémantické podobnosti, kdy se bere v potaz rozdílný význam stejných slov na základě kontextu, ve kterém se vyskytují (*„bat“* v angličtině jako pálka či netopýr). Stejně tak je možné identifikovat dvě různá slova reprezentující stejnou informaci (*„Mary is brave“* má velmi podobný význam jako *„Mary is courageous“*) [14]. Obecným cílem shlukování je rozdělení N bodů v prostoru do K skupin podle vektorů vlastností, kde každý vektor obsahuje soubor vlastností vyjádřených rozměry v multidimensionálním prostoru.

Document clustering se pak snaží dokumenty generující co nejpodobnější vektory shlukovat dohromady. Jsou uváděny dvě metriky využívané ke shlukování - jsou to podobnost, či vzdálenost jednotlivých vektorů. Podobnost a vzdálenost vektorů může být vypočítána

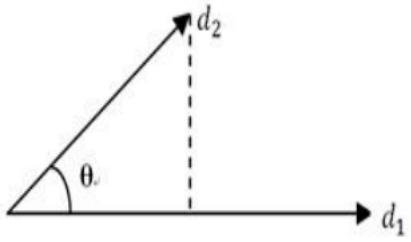
několika různými metodami, jako euklidovská vzdálenost, *Jaccard index* nebo kosinová podobnost [13].

4.1.1 Tvorba vektorů

Ze zkoumaných dokumentů jsou tedy získány relevantní vlastnosti, kde každá z vlastností představuje jeden rozměr multidimensionálního prostoru. Dokument je pak na základě těchto vlastností a dle rozměrů v daných směrech reprezentován jako vektor 4.1:

$$\vec{d} = [w_1, w_2, \ldots, w_n] \tag{4.1}$$

kde w_i je váha nějaké vlastnosti dokumentu (většinou samotný term/slovo). Tato váha pak reprezentuje významnost dané vlastnosti v rámci daného dokumentu. Tento přístup lze demonstrovat například na porovnání dvou dokumentů ve dvourozměrném prostoru, tedy při použití dvou metrik dokumentu. Podobnost se pak například vypočítá jako kosinus úhlu, který tyto dva vektory ve 2D prostoru svírají (obr. 4.1). V reálném případě jsou však vektory n-dimensionální, kde n nejčastěji vyjadřuje počet jednotlivých termů vyskytujících se v daném dokumentu.



Obrázek 4.1: Cosinová podobnost 2 vektorů [17, převzato]

4.1.2 Podobnost dle četnosti výskytů

Jako jedna z vlastností pro zjištění podobnosti je často uváděna metrika *tf-idf (term-frequencies, document-inverse-frequencies)* sloužící pro výpočet váhy slova skrz vícero dokumentů. Přestože se jedná o metodu poměrně triviální, dosahuje celkem dobrých výsledků (vyzkoušeno také prakticky v rámci této práce). Jednotlivé váhy slov pak pro dokument vyjadřují dimense v prostoru (pokud dokument dané slovo obsahuje). Metrika *tf-idf* se skládá ze dvou hlavních složek, kde složka *tf* je vypočítána rovnicí 4.2

$$tf_{t,d} = \frac{n_{t,d}}{\sum_k n_{k,d}} \tag{4.2}$$

kde $tf_{t,d}$ je poměr počtu výskytu slova v dokumentu vůči délce dokumentu, $n_{t,d}$ je počet výskytů slova v dokumentu k a $\sum_k n_{k,d}$ je právě délka dokumentu k. Při této metrice se však do popředí často dostávají slova bez sémantického významu, ale s vysokou frekvencí (v angličtině třeba *the, a*).

Proto má *tf-idf* druhou složku *document inverse frequencies*, která slouží právě pro „zvýhodnění“ těch slov, která se vyskytují s velmi nízkou frekvencí. Předpokládá se totiž, že taková slova mohou mít větší syntaktický význam. *Document inverse frequencies* je tedy vypočítána jako (rovnice 4.3)

$$idf_{t,D} = \log \frac{|D|}{|\{j : t_i \in d_j\}|} \tag{4.3}$$

kde $|D|$ je celkový počet porovnávaných dokumentů v databázi, $|\{j : t_i \ni d_j\}|$ je počet dokumentů obsahující dané slovo a $idf_{t,D}$ je výsledný poměr. Výsledná váha slova je pak spočtena jako (rovnice 4.4)

$$tf\text{-}idf_{t,d,D} = tf_{t,d} \cdot idf_{t,D} \tag{4.4}$$

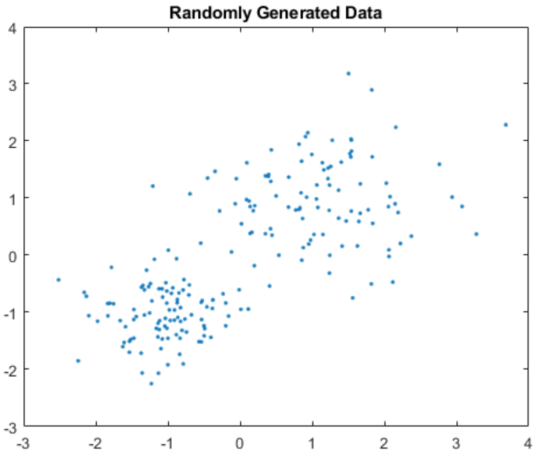
tedy vynásobením dvou výše popsaných složek [27]. Výsledná metrika tedy kombinuje jak váhu termů na základě počtu jejich výskytu v daném dokumentu, tak váhu termů na základě jejich významnosti. V praktických aplikacích a návodech je také možné se často setkat s předzpracováním textu pomocí filtrování takzvaných *stopwords* - čili slov bez žádného sémantického významu. Tímto filtrováním dojde ke zpřesnění složky *term-frequency*, jelikož bude ořezána právě o slova bez sémantického významu.

4.1.3 Sémantická podobnost

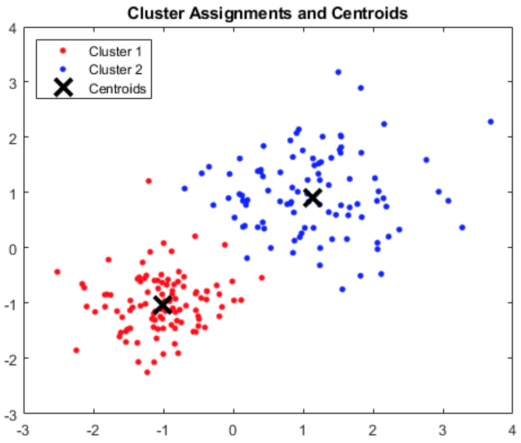
Nádstavbou pro výše uvedený algoritmus je zohlednění polysémie a synonym, jak je zmíněno v úvodu této kapitoly (4.1). Předpokladem je existence lexikální databáze obsahující skupiny různých slov se stejným, nebo velmi podobným významem, případně rozlišuje různé významy jednoho slova. Příkladem existující databáze pro angličtinu je databáze *WordNet*. Stejně tak jako například získávání klíčových slov (3.1.1) je tento postup závislý na jazyku a na vstupní databázi (v případě získávání klíčových slov databáze *stopwords*). Mimo synonyma je dokonce možné nahrazovat zájmena konkrétními jmény osob, pokud se tatáž osoba například vyskytuje ve více větách a je v některé z vět uvozena svým jménem. Tomuto kroku se říká *coreference*. Dalším krokem algoritmu je vyřešení synonym s využitím *WordNet*. Následně je použita analýza dle četnosti výskytu slov, například *tf-idf*. Tuto je možné použít, jelikož v předchozích krocích byly vyjádřeny synonyma a polysémantické termy. Posledním krokem je vytvoření shluků, ať už se jedná o analýzu zohledňující, či ignorující sémantickou podobnost [14].

4.1.4 Tvorba shluků

Existuje celá řada algoritmů pro tvorbu shluků multidimensionálních vektorových prostorech - *K-means, Bisecting K-means, Hierarchical Agglomerative, Particle Swarm Optimization, Self-Organizing Maps*. Z nich je však často akcentován algoritmus *K-means* [14]. *K-means* pracuje na principu iterativního hledání příslušnosti bodů v prostoru do odpovídajících skupin. Zmíněné body nejsou nic jiného než vektory generované pro každý dokument výše popsanými algoritmy. Každou ze skupin je možno reprezentovat takzvaným geometrickým centroidem. Na začátku algoritmu je určeno K takových centroidů (náhodně nebo zadáno), kteří jsou rozmístěni náhodně v prostoru. Každému bodu je následně přirazena příslušnost danému centroidu na základě nejmenší vzdálenosti od daného centroidu. Po přiřazení všech bodů v prostoru jsou pozice centroidů přepočítány tak, aby reprezentovaly centroid skupiny bodů, které jsou k němu přiřazeny. Tento postup je iterativně opakován, dokud nedojde k ustálení bodů a tedy konvergenci algoritmu [19]. Algoritmus je vizualizován na obr. 4.2.



(a) Vstup K-means [1, převzato]



(b) Výstup K-means [1, převzato]

Obrázek 4.2: K-means

4.2 Výpočet vzdálenosti vektorů a určení rozdílnosti dokumentů

V předchozí části této práce byly zevrubně popsány metody pro shlukování dokumentů. Z pohledu této práce však není výhodné vícero dokumentů shlukovat do větších celků, nicméně přesný opak. To však neznamená, že se některé principy popsané výše nedají využít pro některé cíle stanovené touto prací a sice automatizované rozdělování vícero dokumentů spojených do jednoho celku. Hlavním kritériem nebude na rozdíl od shlukování podobnost vícero dokumentů, ale rozdílnost 2 konkrétních částí dokumentu. Navíc tedy není nutné porovnávat velké množství generovaných vektorů současně, ale vždy vektory 2. Toto vychází z předpokladu, že stačí porovnávat rozdílnost vektoru aktuálně zpracovávané stránky dokumentu vůči již zpracované předchozí části. Na základě rozdílnosti aktuální stránky s předchozí částí dokumentu je pak možné určit, jestli se jedná o stránku navazující na předchozí dokument, či se jedná o dokument nový.

Výpočet vzdálenosti vektorů v multidimensionálním prostoru se zdaleka netýká jen shlukování dokumentů. Jedná se o obecný problém, který je zmíněn také například v kapitole o OCR 2.1.4 pro klasifikaci a rozpoznávání znaků. Níže jsou deklarovány nutné předpoklady pro schopnost měření vzdálenosti v prostoru stejně tak jako některé metody pro zjištění podobnosti/rozdílnosti vektorů v multidimensionálním prostoru.

4.2.1 Metrika a metrický prostor

Pro měření vzdálenosti vektorů či bodů v prostoru je nutné mít definováno, co vlastně vzdálenost je a mít určeny základní pravidla, která musí být splněna. Vzdálenost splňující tato daná pravidla se pak nazývá metrika a prostor, ve kterém tato pravidla platí, je nazýván prostorem metrickým. Pro člověka nejpřirozenějším metrickým prostorem je Euklidovský 3-rozměrný prostor. Ten splňuje běžné podmínky symetrie, trojúhelníkové nerovnosti a současně vzdálenost ρ dvou různých bodů musí splňovat $\rho > 0$ [5]. Tedy formálně:

1. $\rho(x, y) \geq 0$
2. $\rho(x, y) = 0 \Leftrightarrow x = y$

3. $\forall x, y : \rho(x, y) = \rho(y, x)$
4. $\forall x, y, z : \rho(x, y) + \rho(y, z) \geq \rho(x, z)$

Tyto podmínky však platí i v n-rozměrných prostorech. Pokud by tyto podmínky nebyly splněny, pohybovali bychom se v jiném než metrickém (euklidovském) prostoru a nemusely by platit klasické předpoklady pro měření vzdálenosti. Následující přístupy k měření vzdálenosti by tedy obecně vzato nebyly univerzálně platné.

4.2.2 Kosinová podobnost

Kosinová podobnost je jednou z metod měření vzdálenosti a počítá s tím, že podobnost dvou dokumentů odpovídá korelaci dvou vygenerovaných vektorů. Tato korelace může být vyjádřena jako úhel svíraný právě těmito vektory přičemž, čím větší je vypočítaný úhel, tím méně jsou si dva dokumenty podobné (obr. 4.1). Mějme tedy dva m-dimensionální vektory reprezentující 2 dokumenty, kde m například označuje počet termů získaných *tf-idf*, poté pomocí rovnice 4.5 vypočítáme právě vzdálenost vektorů.

$$SIM_C(\vec{t_a}, \vec{t_b}) = \frac{\vec{t_a} \cdot \vec{t_b}}{|\vec{t_a}| \times |\vec{t_b}|} \tag{4.5}$$

Korelace je pak vyjádřena v intervalu $(0 - 1)$. Současně tato metrika zanedbává rozdílné délky dokumentů a pokud bychom jednoho dokumentu vytvořili druhý dokument dvojnásobnou kopií, výsledek kosinové vzdálenosti by byl 1, tedy že jsou identické. Což může být ovšem také kontraproduktivní, pokud je pro dané použití i tato rozdílnost důležitá [17].

4.2.3 Ekulidovská vzdálenost

Jedná se o standardní metriku k měření vzdálenosti používanou také v *K-means* algoritmu [17]. Je definována rovnicí 4.6 jako:

$$D_E(\vec{t_a}, \vec{t_b}) = (\sum_{t=1}^{m} |w_{t,a} - w_{t,b}|^2)^{\frac{1}{2}} \tag{4.6}$$

Nejedná se v podstatě o nic jiného, než klasickou vzdálenost dvou bodů v prostoru přirozeném pro člověka, kde tato vzdálenost se počítá jako přepona pravoúhlého trojúhelníku.

4.2.4 Jaccard index

Jaccard index v oblasti určování podobnosti dokumentů zkoumá váhy sdílených a nesdílených slov/termů skrze zoumané dokumenty [17]. Porovnává součet vah slov, která se nacházejí v obou porovnávaných dokumentech (při porovnání dvou dokumentů) vůči součtu vah slov, která se nacházejí právě v jednom ze zkoumaných dokumentů, tedy definováno rovnicí 4.7:

$$I_J(\vec{t_a}, \vec{t_b}) = \frac{\vec{t_a} \cdot \vec{t_b}}{|\vec{t_a}|^2 + |\vec{t_b}|^2 - \vec{t_a} \cdot \vec{t_b}} \tag{4.7}$$

4.3 Obrazová podobnost

Dosud popsané metody zkoumající podobnost dokumentů, či metody k získávání relevantních dat z dokumentů se věnovaly pouze textovým datům a textové reprezentaci zkoumaných dokumentů. Již v úvodu této práce však bylo nastíněno, že je nutné zpracovávat také grafické elementy. Může se však také jednat o celé stránky, ktere nemusí obsahovat text, ale pouze nějakou obrazovou dokumentaci. Tato práce tedy řeší i problematiku podobnosti dvou obrazů. Tato kapitola popisuje několik přístupů k porovnání obrazů a nastiňuje možné problémy a jejich řešení.

4.3.1 Požadavky na obrazovou podobnost

Z kontextu použití výsledné aplikace této práce je evidentní, že se výsledná aplikace musí být schopná vypořádat s uživatelskými vstupy. Tedy se vstupy, které nejsou nijakým způsobem normalizované, či jinak předzpracované. Jelikož je nutné porovnávat jednotlivé stránky takového uživatelského vstupu bez ohledu na intra-třídní variabilitu, je potřebné, aby použité metody pro porovnávání grafické podobnosti byly vůči této variabilitě alespoň částečně invariantní. Pojem invariance je nastíněn v kapitole 2.1.2. Vstupem pro porovnání grafické podobnosti byly v této práci vždy celé stránky dokumentu, nikoliv jen podčásti, jak je to realizováno například při *Document Layout Analysis* a extrakci grafických komponent. Lze tedy použít běžné algoritmy z oboru zpracování obrazu. Vybrané algoritmy jsou popsány v následujících částech práce.

4.3.2 Histogram

Nejobecnějším a nejjednoduším způsobem porovnání dvou obrazových vstupů je histogram. Ten byl také zmíněn v kapitole 2.1.2 věnující se OCR. Pro porovnání histogramů dvou obrazových vstupů existuje několik obecných metod, které jsou používány v knihovnách věnujících se zpracování obrazu. Jsou jimi [3]:

- korelace
- *Chi-square*
- průnik
- *Bhattacharyya distance*

které se liší právě ve způsobu porovnání dvou vstupních histogramů. Přestože je porovnání histogramů opravdu pouze základní technika, jejím použitím bylo později dosaženo zlepšení výsledků aplikace při odhadu dělicích bodů dokumentu. Navíc tato metoda alespoň částečně splňuje požadavky na invarianci - je invariantní minimálně vůči posunutí, zvětšení a rotaci.

4.3.3 SIFT Points

O něco zajímavější a pokročilejší způsob porovnání dvou obrazů je metoda *SIFT Points* neboli *Scale-invariant feature transform*. Metoda byla představena v roce 2004 a celkem se skládá ze 4-5 fází popsaných níže [21].

Scale-space extrema detection

Jak název metody napovídá, je invariantní vůči zvětšení/zmenšení obrazu. Oproti například *Harris corner detector*, který je invariantní vůči rotaci, přidává navíc možnost porovnávat obrazové vstupy jejichž velikost byla také různě transformována. Jádrem toho algoritmu je tedy zanedbání těchto transformací. Toho je dosaženo použitím parametru ρ - takzvaný *scale parameter*. Ten generuje různé velikosti takzvaného okna pro detekci hran a rohů. Doslova se v originální práci uvádí, že jsou prohledávány všechny oblasti přes všechny různé velikosti s využitím efektivního algoritmu *difference-of-Gaussian*. Výstupem této první fáze jsou potencionální klíčové body, které by mohly být porovnávány.

Keypoint localization

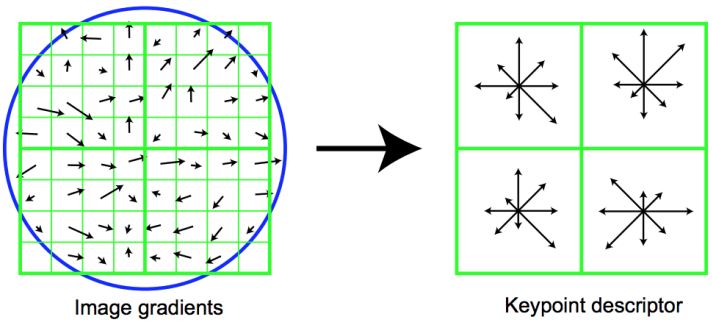
V tomto kroku probíhá výběr klíčových bodů, kde jsou ponechány pouze ty, které splňují určitá kritéria. Tato kritéria se týkají především kontrastu navrhovaných bodů. Pokud je vyhodnoceno, že je kontrast kandidátního bodu příliš malý, je zahozen. Malý kontrast totiž znamená velkou náchylnost k různým druhům zašumění.

Orientation assignement

V této části algoritmu je dosaženo invariance vůči rotaci tak, že každému bodu je přiřazena orientace na základě lokálních vlastností daného bodu. Popisující vlastnosti tohoto bodu se pak váží relativně k danému bodu a jsou invariantní vůči rotaci celého obrázku. Pro přiřazení orientace k bodům je využíván histogram orientací (již vysvětlen v sekci 2.1.2), kde je využíváno 10 různých rozsahů pokrývajících 360°. Přičemž pokud z histogramu vzejde více než jeden významný vrchol histogramu vyjadřující směr orientace (minimálně 80% výšky maximálního vrcholu), je možné jeden bod následně reprezentovat jako více různých bodů s různými orientacemi.

Keypoint descriptor

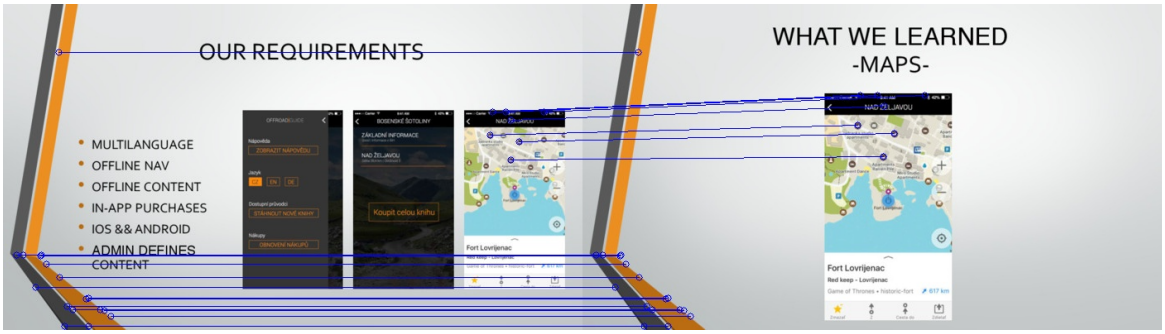
Po získání invariantních bodů vůči rotaci a zmenšení jsou tyto popsány deskriptory. Deskriptor je vytvořen z okolí bodu o rozměrech 16x16 pixelů, kde toto okolí je rozděleno na menší podčásti 4x4 a pro každou podčást je opět vypočítán histogram orientací s přiřazením orientací do 8 kategorií. Celkem existuje tedy 128 hodnot. Z těchto hodnot se pak utvoří vektor, který je využíván jako deskriptor právě daného význačného bodu - *SIFT Point*. Příklad je vidět na obrázku 4.3.



Obrázek 4.3: Histogram orientací při SIFT [21, převzato]

Keypoint matching

V předchozích krocích bylo popsáno získání bodů SIFT. Tyto body se dají samozřejmě získat pro vícero různých obrázků a jak vyplývá z posledního kroku, každý ze získaných bodů je poté reprezentován jako vektor vycházející s histogramu orientací. Je tedy relativně snadné dva obrázky porovnat pomocí těchto vektorů. Pro každý z bodů jednoho obrázku je hledán takzvaný *nearest neighbour* čili nejbližší soused. Blízkost je počítána pomocí Euklidovské vzdálenosti, která byla vysvětlena v části 4.2.3. Porovnání dvou různých obrázků (příklad testovacího vstupu aplikace s malým množstvím textu - prezentace) za pomoci SIFT bodů je ukázáno na obr. 4.4, kde levá a pravá strana jsou dvě různé stránky dokumentu vykresleny vedle sebe pro lepší ilustraci. V tomto konkrétním případě je jen velmi málo markantů, které se dají získat z textu, naopak lze vidět, že porovnání pomocí SIFT dopadlo celkem úspěšně a tyto stránky mají relativně velký překryv.



Obrázek 4.4: SIFT [vlastní]

Dalšími alternativami k SIFT bodům jsou například také body SURF *Speeded Up Robust Features*, nebo takzvaný algoritmus ORB - *Oriented FAST and Rotated BRIEF*, kde se tyto algoritmy snaží buď řešit podobný problém jako SIFT, nebo dokonce přistupují k danému problému podobnou cestou.

4.4 Praktické využití TF-IDF, histogramu a SIFT

Přestože jsou *term-frequencies, document-inverse-frequencies*, histogram a *Scale-invariant feature transform* využívány při řešení naprosto rozdílných problémů, dokonce z různých vědních oborů, nic nebrání tomu tyto rozličné metody zkombinovat a využít dle potřeb. V kontextu této práce je možné je zkombinovat pro porovnávání podobnosti stránek a případně automatizované rozdělování dokumentů. Každá z těchto metod předkládá jinou charakteristiku stránky pro porovnání. Tyto charakteristiky je pak možné například vynést do vektorů a měřit jejich vzdálenosti, využít pro rozhodování na základě váhových koeficientů, či je jinak zpracovat k dosažení daného cíle určení dělícího bodu dokumentu. Konkrétní praktické využití je pak popsáno v další kapitole 5.

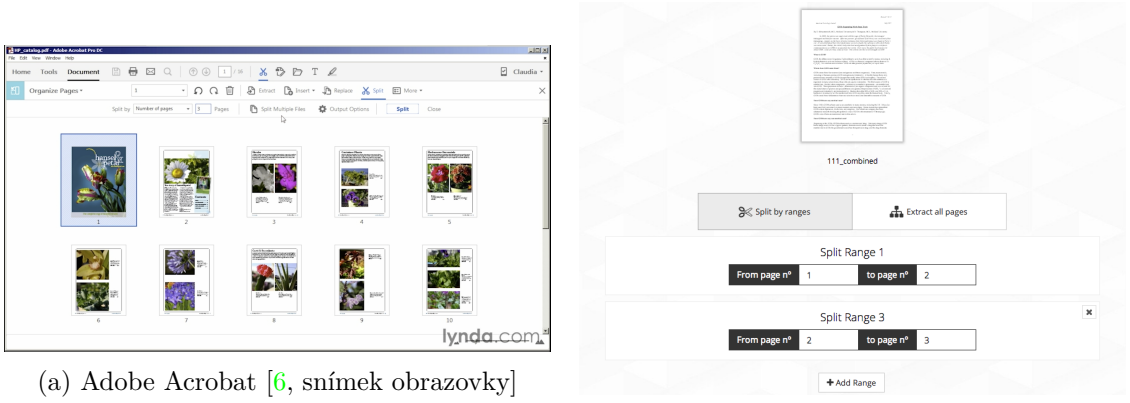
V této části práce bylo nastíněno několik způsobů k porovnání dokumentů na základě textových informací získaných z dokumentu, ale také na základě obrazové analýzy. Kromě obecných principů porovnávání vícero dokumentů a jejich shlukování se kapitola zabývá konkrétními metodami jako *tf-idf*, sémantickou analýzou a měřením vzdáleností mezi vektory reprezentujícími jednotlivé dokumenty. Ve druhé části kapitoly jsou poté popsány dvě metody použité v aplikační části této práce pro porovnávání obrazové podobnosti.

Kapitola 5

Realizace aplikace

Kromě prozkoumání existujících metod pro automatizované rozdělování dokumentů a získávání informací z dokumentů bylo jedním z hlavních cílů této práce vytvořit prototyp aplikace, která získané teoretické informace prakticky využije a bude umožňovat právě automatizované rozdělování a získávání relevantních dat.

Před samotnou implementací je však vhodné zevrubně prozkoumat existující řešení zabývající se podobnou problematikou. Existuje nespočet nástrojů a prohlížečů dokumentů, které nabízejí rozdělování dokumentů. Z uživatelského hlediska je k tomuto problému přistupováno několika způsoby. Nejčastěji je možno si zvolit fixní počet stránek po kolika bude dokument rozdělen (například Adobe Acrobat, obr 5.1a). Dále je také možno manuálně vypsat seznam stránek jako začátky a konce jednotlivých subdokumentů (online nástroj - obr. 5.1b), nebo extrahovat stránky takzvaným *drag&drop* - tedy přetažením jednotlivých stránek. Ačkoliv jsou mnohé tyto nástroje velmi intuitivní, žádný z nich nenabízí plně automatizované rozdělování dokumentů dle obsahu a vždy je tato práce ponechána jen na uživateli a je vyžadována jeho manuální práce. Ve zmíněných nástrojích také zcela chybí možnost automatizované anotace dokumentů. Tu zase nabízejí jiné nástroje, které však neumožňují jejich rozdělování. A právě kombinace automatizovaného rozdělování a extrakce relevantních dat je jedním z hlavních cílů implementované aplikace. Její algoritmus, implementované postupy a využití dříve získaných informací a metod vysvětlených v předchozích částech této práce jsou popisovány v této kapitole.



(a) Adobe Acrobat [6, snímek obrazovky]

(b) Online nástroj [2, snímek obrazovky]

Obrázek 5.1: Existující nástroje

5.1 Specifikace požadavků na aplikaci

Kromě dvou hlavních požadavků rozdělování dokumentů dle obsahu a získávání informací z dokumentů existuje celá řada doprovodných (uživatelských či vývojářských) předpokladů pro úspěšnou realizaci aplikace a jednoduchou údržbu. Mezi tyto požadavky patří především:

- vybrání souboru k analýze
- analýza nahraného souboru k získání návrhu automatizovaného rozdělení
- potvrzení návrhu automatizovaného rozdělení
- opravení špatně navržených dělicích bodů dokumentu
- opravení špatně získaných informací z dokumentu
- uložení analyzovaného a rozděleného dokumentu pro pozdější práci
- vynesení dokumentů na časovou osu se zobrazením získaných relevantních dat a možností jejich úpravy + možnost úpravy časové osy
- nastavení informací, které si uživatel přeje o dokumentu/dokumentech získat automatizovanou analýzou
- export a stažení jednotlivých dokumentů (i více současně)
- export a stažení obrazové reprezentace časové osy

Dalšími funkčními požadavky na aplikaci jako takovou jsou:

- automatické asnychronní ukládání rozpracované úpravy dat pro zamezení jejich ztráty
- oddělený backend a frontend aplikace pro možnost použití aplikační logiky nezávisle na uživatelském rozhraní
- moderní uživatelské rozhraní podporující asynchronní úpravy dat

Tyto požadavky vycházejí z především předpokládaného použití aplikace z uživatelského pohledu. Pro relizaci aplikace je možné zvážit několik přístupů a několik různých architektur. Od desktopové, přes mobilní až po webovou aplikaci. Nicméně na základě výše uvedených požadavků se zdá býti vhodné postavit aplikaci s webovým rozhraním. Toto má několik dalších výhod, jsou jimi:

- oproti desktopové verzi o mnoho jednodušší distribuce uživatelům
- dostupnost dat kdekoliv
- z pohledu uživatele využití externího úložiště a ušetření místa na disku
- z pohledu uživatele není nutný velký výpočetní výkon

5.2 Architektura aplikace a zvolené technologie

Moderních přístupů k vývoji webových aplikací je velké množství. Ať už se jedná o základní architektonické přístupy (*Model-View-Controller*), MVVM (*Model-View-ViewModel*) a podobné, či o přístup k řešení škálovatelnosti moderní aplikace, například pomocí takzvaných *microservices*, mají jedno společné. Jsou založeny na tvorbě podčástí aplikace tak, aby byly na sobě nezávislé a podporují komunikaci jednotlivých nezávislých podčástí aplikace přes takzvaný *Application Programming Interface*, zkráceně *API*. Tento přístup nejenže podporuje přirozenou cestou tvorbu architektury aplikace, ale také má vliv na její škálovatelnost. Z tohoto pohledu se dá implementovaná aplikace rozdělit na 3 hlavní části. Mezi ně patří samotná aplikační logika porovnávající podobnost jednotlivých podčástí dokumentu, backend aplikace starající se o přidružené operace jakožto ukládání, komunikaci s uživatelem a podobné. Nakonec část vizualizační, která je naprosto nezávislá na samotné logice aplikace.

5.2.1 Aplikační logika

Pro realizaci části aplikace, která má na starost porovnávání podobnosti jednotlivých podčástí dokumentů, návrh dělicích bodů a získávání relevantních informací z dokumentů, se přímo nabízí programovací jazyk *Python*. Ten nabízí širokou škálu knihoven pro vědecké použití, které velmi usnadňují i jinak velmi složité operace a výpočty. Konkrétně byly v této práci mimo jiné použity knihovny *numpy*, *sklearn* a *opencv*. Celkem byly implementovány 3 různé moduly:

- modul pro výpočet *tf-idf* metriky vstupních podčástí dokumentu
- modul pro výpočet podobnosti histogramů dvou stránek dokumentu
- modul pro výpočet grafické podobnosti na základě *SIFT Points*

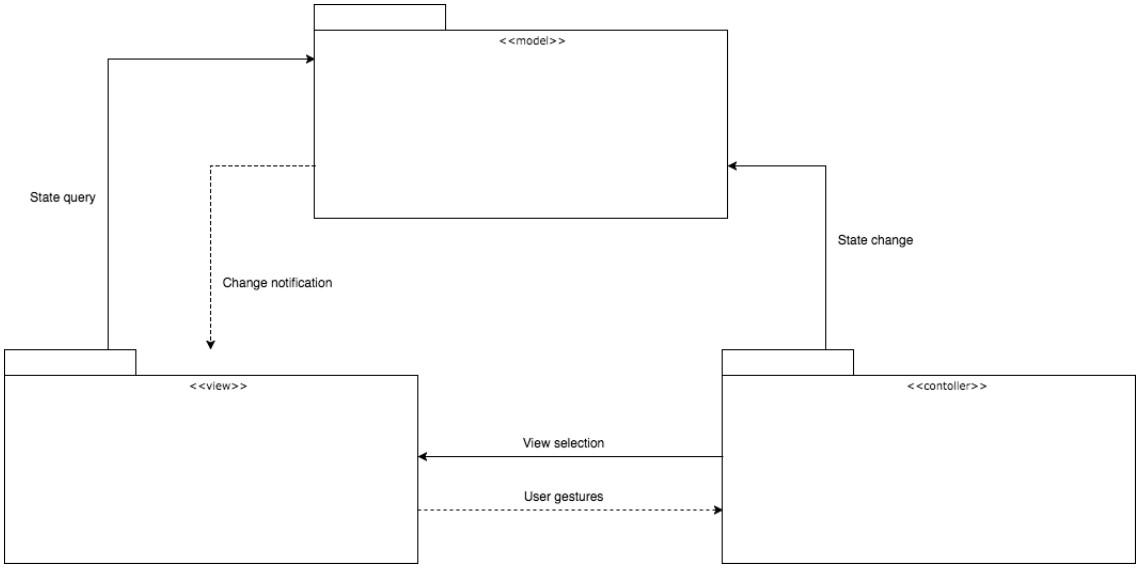
Každá z implementovaných částí je naprosto nezávislá na částech ostatních. Je tedy jednoduše možné na zadaný dokument aplikovat všechny, nebo pouze některé z metrik, které tyto moduly generují. Výstupem každého modulu je pak metrika vyjadřující podobnost/rozdílnost zadaných vstupních dat, která je vypočítána na základě popsaných metod v kapitolách 4.2 a 4.3.

5.2.2 Backend aplikace

Backend aplikace tvoří v této práci páteřní část, která má na starost jak přípravu vstupních dat, tak volání výše popsaných modulů a zpracovávání předložených výsledků. Tato hlavní část byla implementována v *PHP* frameworku *Laravel*. Jedná se o moderní framework postavený na zmíněné architektuře *MVC*. Ta je rozdělena na 3 hlavní části, ty jsou vyobrazeny na obr. 5.2 a jsou také popsány v textu níže.

Model

Model reprezentuje doménové informace a stará se o takzvanou *business logic* aplikace. Ve frameworku Laravel existuje předchystaný model, který implementuje všechny potřebné konstruktory, *magic methods* a další. Ten je nicméně pevně svázán s ukládáním modelových dat do databáze (podporuje samozřejmě více druhů databází). Při uživatelském nahrávání souboru k analýze by pak bylo potřeba jej svázat s daným uživatelem. To by vyžadovalo



Obrázek 5.2: Model-View-Controller [vlastní]

registraci a další režii. Navíc cílem aplikace je práce s velkými soubory a generování velkého množství dalších podčástí dokumentu - dalších souborů. To vyžaduje systém pro správu souborů na disku a do databáze by nakonec ukládalo jen velmi malé množství dat. Z tohoto důvodu byl implementován vlastní model založený na použití *sessions*. *Session* je globální proměnná, která po stanovenou dobu udržuje stav aplikace, není tedy problém za pomoci *sessions* ukládat stavová data o zpracovávání jednotlivých dokumentů, ani výsledky analýzy.

View a Controller

Controller neboli řadič je část aplikace, která má na starost zpracování uživatelských požadavků z *routes* (seznam definovaných url adres), volá požadované metody z modelu, který provádí operace nad samotnými daty a standardně vrací nový *View*, který reprezentuje aktualizovaná data uživateli. Jedním z přístupů modernějších aplikacích je však celkové oddělení této vrstvy a dotazování na aktualizaci dat probíhá asynchronně například za pomoci klientského javascriptu. Výměna dat mezi modelem, řadičem a uživatelským rozhraním pak probíhá práve přes API.

5.2.3 Vizuální část aplikace a API

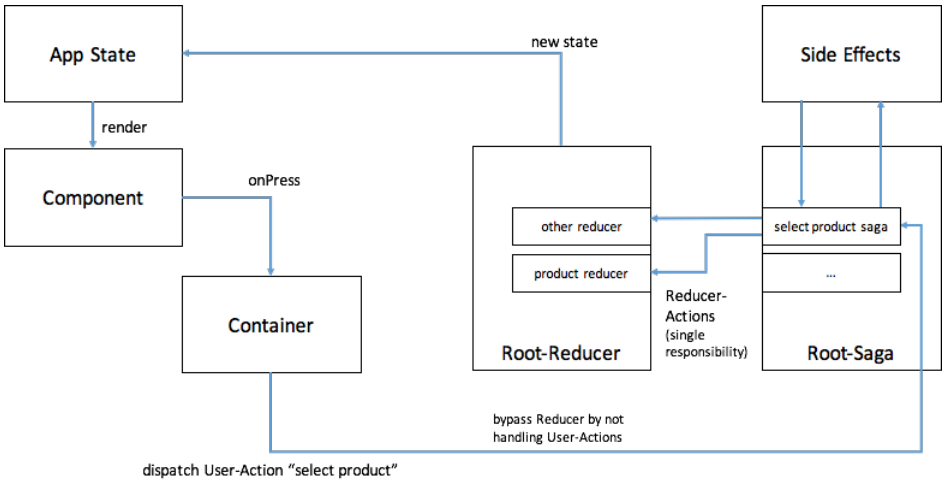
Existují tedy v zásadě 2 způsoby, jak prezentovat informace uživateli ve *View* v rámci webových aplikací. Jsou to *server-side rendering* a *client-side rendering*. V rámci této práce byly zkombinovány oba dva přístupy. Nejprve je z kontroleru vrácen vždy stejný *View* obsahující statické informace (renderování ze serveru) a element, do kterého se následně injektují dynamicky vytvářené prvky HTML (renderování na straně klienta). Na klientské straně k tomu byl využit moderní javascriptový framework *React* v kombinaci s podpůrnými knihovnami pro udržování globálního stavu aplikace (*Redux*) a pro asynchronní komunikaci přes API (*Sagas*).

React, Redux, Sagas

Pro prezentaci informací uživateli ve *View* neboli pro vytvoření takzvaného *frontedu* byla tedy využita kombinace 3 nástrojů, kdy *React* se stará o *frontend* logiku. V rámci *React framework* se hovoří o 2 hlavních entitách, kterými jsou *component* a *container*. Tyto dvě entity se liší přístupem ke globálnímu stavu pro *frontend* aplikace, kdy komponenta udržuje pouze svůj vnitřní stav a nesdílí ho se svým okolím, není schopná číst stav okolních komponent či stav globální. Naproti tomu kontejner je napojen na globální stav aplikace či stavový kontejner (knihovna *Redux*), kdy je schopná tento globální stav číst, nebo i upravovat přes takzvaný *dispatcher* - toto už je však pojem spjatý přímo s *Redux*.

Ten udržuje neměnný *immutable* stav, kdy se při zavolání akce přes *dispatcher* aktuální stav neupravuje, nýbrž je vrácen stav nový se zohledněním žádaných změn. V rámci *Redux* se také v žádném případě neprovádí žádné akce jako stahování dat ze serveru, úprava dat na serveru a podobné. Jedná se tedy o kolekci čistých *pure* funkcí, které nemají takzvané *side-effects*. Toto paradigma vychází z funkcionálního programování.

Naopak *Sagas* se starají právě o zmíněnou komunikace se serverem, o zasílání dat a o řešení neočekávaných stavů. Typickým příkladem je dotaz na server, kdy ten může skončit jakoukoliv chybou a dopředu se nedá říci, co bude výsledkem. Navíc může trvat různě dlouhou dobu. *Sagas* tedy přes takzvaný *middleware* zachytávají akce volané kontejnery přes *dispatcher* určené pro *Redux* a pokud má mít akce charakter komunikace se serverem, *Sagas* právě tuto komunikaci řeší (na rozdíl od staršího přístupu javascriptu přes tzv. *promises* je využíváno novější funkcionality, která přišla se standardem ES6 - generátory). Po odpovědi ze serveru pak může být opět změněn globální stav aplikace a uživateli zobrazena výsledná data. Diagram celkové spolupráce těchto 3 částí je na obr. 5.3.



Obrázek 5.3: React, Redux, Sagas [7, převzato]

REST API

V rámci návrhu API byla použita architektura *REST - Representational State Transfer*, která je velmi úzce spjata s protokolem *HTTP*, kdy *REST* odráží jak jeho bezstavovost, tak také použité metody (například *get, put, post, delete, ...*). Celkem je v aplikaci používáno 11 koncových bodů (*endpoints*) pro komunikaci klientského javascriptu (*React*) s backendem aplikace (*Laravel*). Tyto body jsou použity pro nahrávání souboru k analýze, pro získání ná-

vrhu dělicích bodů, jejich úpravu, získání relevantních dat, úpravu relevantních dat, úpravu časové osy a stažení zkoumaných dokumentů.

Z výše popsaného však vyplývá, že architektura *REST* nebyla implementována ve své čisté formě, nicméně byla porušena její podmínka bezstavovosti. To je způsobeno použitím *sessions* na backendu aplikace. Jinými slovy, koncový bod vrací různé výsledky podle aktuálního stavu aplikace na serveru - například při prvním načtení stránky se liší odpověď v případě existujícího rozpracovaného uživatelského projektu a v případě neinicializované *session*. To však v žádném případě nebrání praktickému použití aplikace.

Podpůrná infrastruktura

Pro výsledný prototyp aplikace byl také zřízen webový server na veřejně přístupné adrese http://89.221.216.63/splitter/. Na serveru je nainstalován operační systém Debian 8. Webový server je pak obsluhován pomocí Apache2, na kterém běží PHP verze 7.2. Kromě těchto nástrojů je možno na serveru také nalézt Python 2.7 (Scikit, Scipy, Opencv, Numpy), Laravel 5.3 a NodeJs. Tyto aplikace a knihovny byly nutné k implementaci či překladu zdrojových kódů. Dalšími podpůrnými nástroji jsou Tesseract, Ghostscript, Imagick a Pdftk.

Vizuální prvky

Celý *frontend* webové aplikace je postaven na přístupu *Material Design*. Jedná se o přístup k tvorbě uživatelských rozhraní, který klade důraz na funkčnost jednotlivých prvků a jejich minimalistické vyobrazení. Konkrétně pak byla použita knihovna *material-ui*, která disponuje velkým množstvím předchystaných elementů. Vizuální část práce je pak přiložena v příloze A.

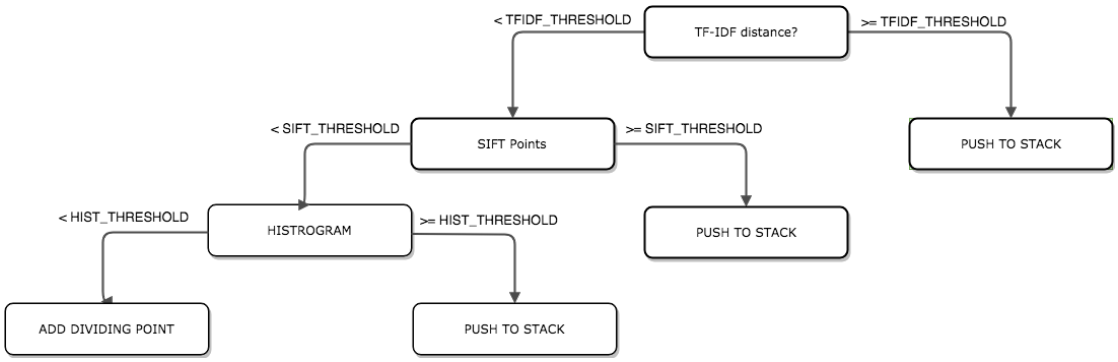
5.3 Implementovaný algoritmus

V této části práce je popsán algoritmus implementovaný vytvořenou aplikací. Skládá se z několika přípravných kroků a samotné analýzy. Program obecně očekává na vstupu soubor ve formátu pdf. Ten je rozdělen na jednotlivé stránky, kdy z každé stránky je poté získán text. Jak bylo zmíněno v kapitole 2.2, OCR je velmi časově náročné. Proto se systém nejprve snaží získat textová data jiným způsobem - analýzou metadat souboru a pouze v případě, že se toto nepodaří, je spuštěna analýza OCR (Tesseract).

Po získání textových dat jsou pak analyzovány stránky na svou podobnost. Původním záměrem bylo získat o každé jednotlivé stránce co nejvíce informací, ty vynést do multidimensionálního vektoru a porovnat vzdálenosti jednotlivých vektorů. Tento systém se však ukázal jako ne příliš efektivní, jelikož nezohledňoval dokument jako celek, nýbrž porovnával vždy jen jeho poslední stránku (pokud se bere v potaz obrazová podobnost). Aktuálně je tedy používán zásobník stránek, který se postupně plní stránkami, které jsou vyhodnoceny jako jeden dokument. V případě, že následující stránka nesplní stanovená kritéria, je zásobník označen za finální dokument, vyprázdněn a toto se celé opakuje. Při tomto způsobu však není možné použít porovnávání stanovených kritérií pomocí vektorů. Na zásobník se sice postupně mohou přidávat textová data a neustále zpřesňují obsah dokumentu, nicméně obrazová podobnost se dá získat pouze pro dvě entity. Každá metrika tedy slouží jako samostatný markant a tyto markanty jsou pak porovnány nezávisle. V praxi to znamená, že při porovnání stránky vůči zásobníku jsou sice pomocí metody *tf-idf* interně informace reprezentovány právě jako vektory v multidimensionálním prostoru a je spočítána jejich

vzdálenost jako cosinová podobnost, ale tato informace je systému i ostatním metrikám zcela skryta. Pouze vrací výsledek reprezentující vzdálenost vektorů. Metody zjišťující obrazovou podobnost tuto pak zjišťují právě pouze pro poslední a aktuálně zpracovávanou stránku.

I při tomto způsobu vyhodnocování dokumentu jako celku s přihlédnutím k obrazové podobnosti poslední stránky vůči zpracovávané však existuje několik způsobů vyhodnocení výsledků. Prvním z implementovaných způsobů bylo použití rozhodovacích stromů. Navrhovaný strom je na obr 5.4.



Obrázek 5.4: Rozhodovací strom [vlastní]

Zde se však ukázal problém najít vhodné nastavení prahů pro rozhodnutí na jednotlivých úrovních. Z povahy stromu má největší váhu koncový uzel - i pokud by předchozí dva rozhodly ve prospěch dělícího bodu, poslední uzel by toto rozhodnutí vždycky mohl zvrátit. Řešením by bylo správné nastavení prahů, to se však ukázalo jako velmi problematické a při více různých vstupech byly výsledky rozporuplné. Nakonec tedy zvítězila jednoduchá hlasovací varianta, kdy musí být většinová shoda na dělicím bodu.

Po analýze dělících bodů jsou pak pro každý dokument získána relevantní data pomocí *Aylien API*, přičemž důraz je kladen především na osoby, lokace, klíčová slova. Výsledky *Ayilen* při získávání datumů jsou však tristní. Proto byl implementován vlastní regulární výraz, který datumy získává z textu dokumentu.

Jednotlivé dokumenty jsou pak vyneseny na časovou osu, kde uživatel může získané údaje libovolně upravovat, může dokumenty libovolně řadit, může volit, které informace zobrazit, může zobrazovat i stahovat jak jednotlivé dokumenty, tak vícero současně a také je schopen uložit obrazovou reprezentaci časové osy. Implementovaný postup je také popsán algoritmem 1.

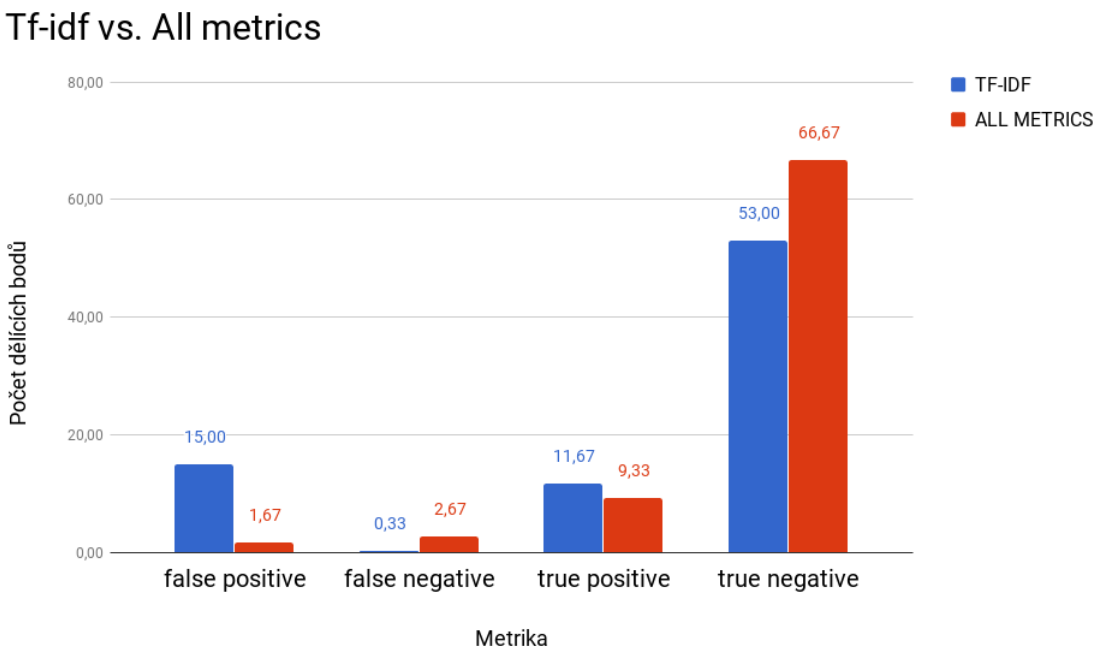
```
Data: Multiple documents merged into one document
Result: Timeline of splitted documents
initialization;
split pdf document by pages;
foreach page in pages do
    if have text within analyzed page then
        save text;
    else
        OCR page using Tesseract;
        save text;
    end
end
create empty stack;
while have unprocessed pages do
    votes = 0;
    compare tf-idf of actual page to stack;
    analyze result and update votes;
    compare graphical similarity of current page to last page pushed to stack;
    analyze result and update votes;
    compare histogram of current page to stack last page pushed to stack;
    analyze result and update votes;
    if votes > threshold then
        push page into the stack;
    else
        save stack as new document;
        empty stack and push actual page to stack;
    end
end
show suggested split points;
get user feedback and process possible corrections;
foreach document in documents do
    mine important data such as persons, locations, dates (Ayilen API + regex);
end
create timeline from splitted documents with mined information;
get user feedback and process possible corrections;
```

Algoritmus 1: Implementovaný algoritmus [vlastní]

5.4 Testování implementovaného prototypu a hodnocení úspěšnosti

Pro hodnocení úspěšnosti implementovaného prototypu bylo použito několik synteticky spojených dokumentů obsahujících jak stránky obsahující text, tak stránky skenované, nebo stránky bez jakéhokoliv rozpoznatelného textu. Také bylo použito několik dokumentů z oblasti práva, kde jsou volně dostupné takzvané *mock cases*, například *Davis vs. Happyland Toy Company*. Tyto obsahují velké množství spojených dokumentů do jednoho celku a tyto dokumenty jsou velmi různých typů jako emaily, ručně psané poznámky, fotky, výpovědi svědků a podobné. Celkem bylo tedy otestováno několik tisíc jednotlivých stránek. Pro hodnocení úspěšnosti byly použity koncepty *False positive, False negative, True positive a True negative*. Ty vyjadřují chybné přijetí hypotézy, chybné odmítnutí hypotézy, validní přijetí hypotézy a validní odmítnutí hypotézy [10]. V kontextu implementované aplikace se za hypotézu považuje návrh dělicího bodu dokumentu. *False positive* je tedy chybně navržen dělicí bod dokumentu v místě, kde by neměl být a kde k dělení nemá dojít. *False negative* je naopak chybějící dělicí bod v místě, kde jeden dokument končí a další začíná. Zbývající *True positive* a *True negative* pak označují správně navržen, či nenavržen dělící bod dokumentu.

Při měření úspěšnosti byly také porovnávány různé nastavení implementovaného systému - například byla porovnávána výkonnost pouze *tf-idf* s výkoností kompletního systému zahrnujícího nejen *tf-idf*, ale také *SIFT Points* a *Histogram*. Toto porovnání je vyobrazeno na obr. 5.5, kde hodnoty počtu dělicích bodů jsou průměrované hodnoty pro vícero zkoumaných dokumentů.



Obrázek 5.5: Hodnocení úspěšnosti [vlastní]

V grafu lze vidět jednoznačný rozdíl mezi použitím samotné *tf-idf* a použitím v kombinaci s obrazovou podobností. Zásadní vylepšení lze vidět na *False positive* - to tedy znamená,

že obrazová podobnost vylepšila vyhodnocování souvislosti dokumentů kdy se výrazně snížil počet špatně navržených dělicích bodů. Při *tf-idf* byl tento v průměru dokonce vyšší než počet správně navržených dělicích bodů. Konkrétně se jednalo o 56.25 %, kdežto při kombinaci vícero metrik se výskyt *False positive* snížil na 15.15 % tedy o více než 40 %. Naproti tomu lze pozorovat drobné zhoršení u *False negative* - tedy některé dělicí body chybí na místech, kde by se měly vyskytovat. Tento trend jde ruku v ruce se snížením výkonnosti na *True positive*, jelikož tyto dva koeficienty jsou pevně provázány. Posledním koeficientem je pak *True negative*, který se při použití kombinace metod dočkal také zlepšení, podobně jako koeficient *False positive*.

Celková úspěšnost implementovaného prototypu systému je tedy pak:

- 77.7 % z pohledu správně nalezených a chybějících dělicích bodů
- 97.5 % z pohledu správně vynechaných a redundantních dělicích bodů

Z výše uvedeného tedy lze vidět, že systém velmi pravděpodobně odhalí konec jednoho a začátek dalšího dokumentu s přesností 77.7 %. Dále lze také uvést, že systém navrhne špatně dělicí bod rozdělující jeden dokument na více částí pouze ve 2.5 % případů.

Kapitola 6

Závěr

V práci jsou uvedeny klíčové aspekty nutné k implementaci prototypu systému pro automatizované rozdělování dokumentů a získávání relevantních dat z dokumentů. V první části práce byl rozebrán především teoretický základ nutný pro tvorbu takové aplikace, jmenovitě pak rozpoznávání textů z obrázků a ze skenů dokumentů pomocí OCR, získávání relevantních informací z dokumentů a metody sloužící k měření podobnosti dokumentů. Další část práce pak byla věnována praktické implementaci systému využívajícího zkoumané metody a přístupy.

V kapitole zabývající se OCR byly popsány obecné principy *Optical Character Recognition*, jednotlivé klasifikační vlastnosti sloužící pro rozpoznávání znaků, jejich zpracování, byly uvedeny příklady existujících systémů a bylo popsáno praktické využití v této práci. Zajímavým pokračováním této části by mohlo být porovnání výkonnosti jednotlivých existujících OCR systémů a jejich následná kombinace pro dosažení co nejlepších výsledků.

V další kapitole věnující se získávání relevantních dat z dokumentů byly rozebrány metody pro získání klíčových slov daných dokumentů (především metoda *Rapid Keyword Extraction*) a také metody zabývající se takzvaným *Named Entity Recognition*. Zde byly vysvětleny principy stojící za získáváním entit jako osob, lokací, datumů a dalších z analyzovaných dokumentů, především pak *Hidden Markov Models*. Závěrem bylo popsáno praktické použití existujících systémů pro praktickou část této práce. Již v této kapitole byla zmíněna možná rozšíření pro zkoumání vztahů mezi jednotlivými entitami.

Jednou z důležitých kapitol této práce je také část věnující se porovnávání podobnosti dokumentů. V té byly popsány principy z oboru *Document Clustering* - například algoritmus pro zjištění podobnosti dle četnosti výskytů (*TF-IDF*) nebo obecné shlukovací algoritmy typu K-means. Současně zde bylo také zmíněno možné rozšíření analýzy podobnosti na základě sémantického významu jednotlivých slov za použití lexikálních databází. V návaznosti na podobnost a shlukování dokumentů pak byly zkoumány obecné metody pro měření podobnosti nejen dokumentů, ale jakýchkoliv entit pomocí vektorů vlastností a především tedy metody měření vzdálenosti těchto vektorů. Dále byly nastíněny metody pro porovnání obrazové podobnosti jednotlivých podčástí zkoumaných dokumentů - tyto slouží v této práci především pro automatizované rozdělování. Dopodrobna byl vysvětlen algoritmus pro získání takzvaných *SIFT Points*, využití histogramu a další.

Závěrem byla popsána praktická implementace prototypu systému pro automatizované rozdělování dokumentů a získávání relevantních dat. Nejdříve byly specifikovány obecné požadavky na takovou aplikaci jak z uživatelského, tak z programátorského hlediska. Poté byly popsány zvolené technologie a architektura. Speciálně pak architektonické přístupy jako *MVC*, principy *REST API*, implementace jednotlivých podčástí včetně použitých ná-

strojů a jazyků (Python, Laravel, React) a podpůrná infrastruktura. V neposlední řadě byl dopodrobna vysvětlen algoritmus implementovaného prototypu a vyhodnocena úspěšnost systému, která se pohybuje mezi 77.7 % a 97.5 % dle zkoumané veličiny. K této části práce se nabízí hned několik možných rozšíření. Zajímavé by bylo sledovat změny úspěšnosti systému při použití vícero metrik, například při vyhodnocení rozložení prvků na stránce, speciálně se pak například pokusit detekovat opakující se záhlaví, či čísla stránek, nebo využít informace získané pomocí *NER*. Dalším možným tématem k prozkoumání pak mohou být jiné způsoby kombinace získaných informací pro navržení dělicích bodů - zde byla diskutována možnost reprezentace jednotlivých podčástí dokumentů jako vektorů kombinujících všechny získané informace, ale nakonec je tento způsob využíván jen částečně. Zajímavé by bylo jistě také použití neuronových sítí a strojového učení a porovnání výsledků s implementovanou heuristikou.

Literatura

[1] Matlab Documentation K-means Clustering. [Online; navštíveno 24. 4. 2018].
URL https://www.mathworks.com/help/stats/kmeans.html

[2] Online nástroj ilovepdf.com. [Online; navštíveno 15. 5. 2018].
URL https://www.ilovepdf.com/split_pdf

[3] OpenCV Documentation CompareHist. [Online; navštíveno 24. 4. 2018].
URL https://docs.opencv.org/2.4/modules/imgproc/doc/histograms.html?highlight=comparehist#comparehist

[4] Přednáška předmětu Modelování a Simulace na FIT VUTBR. [Online; navštíveno 23. 4. 2018].
URL https://wis.fit.vutbr.cz/FIT/st/course-files-st.php?file=%2Fcourse%2FIMS-IT%2Flectures%2FIMS-4.pdf&cid=10316

[5] Skripta předmětu Matematické struktury v informatice na FIT VUTBR. [Online; navštíveno 11. 1. 2018].
URL https://wis.fit.vutbr.cz/FIT/st/course-files-st.php?file=%2Fcourse%2FMAT-IT%2Ftexts%2FZaklady_funkcionalni_analyzy_opr.pdf&cid=11539

[6] Split a document - Adobe Acrobat. [Online; navštíveno 15. 5. 2018].
URL https://helpx.adobe.com/acrobat/how-to/split-pdf-document.html

[7] Using React (-Native) with Redux and Redux-Saga. [Online; navštíveno 24. 4. 2018].
URL https://medium.com/@marcelschulze/using-react-native-with-redux-and-redux-saga-a-new-proposal-ba71f151546f

[8] Abraham, A.; Das, S.; Konar, A.: Document Clustering Using Differential Evolution. In *2006 IEEE International Conference on Evolutionary Computation*, 2006, ISSN 1089-778X, s. 1784–1791, doi:10.1109/CEC.2006.1688523.

[9] Aylien: *Text Analysis API Demo*. developer.aylien.com, Říjen 2017, [Online; navštíveno 22. 12. 2017].
URL https://developer.aylien.com/text-api-demo?text=https%3A%2F%2Fen.wikipedia.org%2Fwiki%2FNamed-entity_recognition&language=en&tab=entities

[10] Banerjee, I.; Bhadury, T.: Self-medication practice among undergraduate medical students in a tertiary care medical college, West Bengal. *Industrial Psychiatry Journal*, ročník 18, č. 2, 2009: s. 127–131, doi:10.4103/0022-3859.97175.

[11] Berry, M. W.; Kogan, J.: *Text mining:applications and theory*. Chichester, U.K.: Wiley, 2010, ISBN 978-0-470-74982-1.

[12] Cheriet, M.; Kharma, N.; lin Liu, C.; aj.: *The Character recognition systems: a guide for students and practioners*. Wiley-Interscience, 2007, ISBN 978-0-471-41570-1.

[13] Chim, H.; Deng, X.: Efficient Phrase-Based Document Similarity for Clustering. *IEEE Transactions on Knowledge and Data Engineering*, ročník 20, č. 9, Sept 2008: s. 1217–1229, ISSN 1041-4347, doi:10.1109/TKDE.2008.50.

[14] Desai, S. S.; Laxminarayana, J. A.: WordNet and Semantic similarity based approach for document clustering. In *2016 International Conference on Computation System and Information Technology for Sustainable Solutions (CSITSS)*, Oct 2016, s. 312–317, doi:10.1109/CSITSS.2016.7779377.

[15] Duan, D.; Xie, M.; Mo, Q.; aj.: An improved Hough transform for line detection. *2010 International Conference on Computer Application and System Modeling (ICCASM 2010)*, 2010: s. V2–354–V2–357, doi:10.1109/ICCASM.2010.5620827.

[16] Hu, M.-K.: Visual pattern recognition by moment invariants. *IEEE Transactions on Information Theory*, ročník 8(2), 1962: s. 179–187, ISSN 0018-9448, doi:10.1109/TIT.1962.1057692.

[17] Huang, A.: Similarity Measures for Text Document Clustering. doi:10.1.1.332.4480, [Online; navštíveno 11. 1. 2018].
URL http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.332.4480&rep=rep1&type=pdf

[18] Janovič, T.: *Detekce polohy očí v obraze obličeje pomocí Houghovy transformace*. bakalářská práce, Brno: Vysoké Učení Technické v Brně, Fakulta Elektrotechniky a Komunikačních Technologií, 2010.

[19] Kanungo, T.; Mount, D. M.; Netanyahu, N. S.; aj.: An efficient k-means clustering algorithm: analysis and implementation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, ročník 24, č. 7, Jul 2002: s. 881–892, ISSN 0162-8828, doi:10.1109/TPAMI.2002.1017616.

[20] Keyes, L.; Winstanley, A.: Using moment invariants for classifying shapes on large-scale maps. *Computers, Environment and Urban Systems*, ročník 25, 2001: s. 119–130, ISSN 01989715, doi:10.1016/S0198-9715(00)00041-7.

[21] Lowe, D. G.: Distinctive Image Features from Scale-Invariant Keypoints. *International Journal of Computer Vision*, ročník 60, č. 2, Nov 2004: s. 91–110, ISSN 1573-1405, doi:10.1023/B:VISI.0000029664.99615.94.
URL https://doi.org/10.1023/B:VISI.0000029664.99615.94

[22] Mori, S.; Suen, C. Y.; Yamamoto, K.: Historical review of OCR research and development. *Proceedings of the IEEE*, ročník 80, č. 7, 7 1992: s. 1029–1058, ISSN 0018-9219.

[23] Morwal, S.: Named Entity Recognition using Hidden Markov Model (HMM). *International Journal on Natural Language Computing*, ročník 1, č. 4, 2012-12-31: s. 15–23, ISSN 23194111, doi:10.5121/ijnlc.2012.1402.

[24] O'Gorman, L.; Kasturi, R.: *Executive briefing: document image analysis*. IEEE Computer Society Press, 1997, ISBN 08-186-7802-X.

[25] Podobny, Z.: *Improving the quality of the output*. GitHub, Říjen 2017, [Online; navštíveno 19. 12. 2017].
URL https://github.com/tesseract-ocr/tesseract/wiki/ImproveQuality

[26] Rabiner, L. R.: A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, ročník 77, č. 2, Feb 1989: s. 257–286, ISSN 0018-9219, doi:10.1109/5.18626.

[27] Rajaraman, A.; Ullman, J. D.: *Mining of Massive Datasets*. Cambridge University Press, 2011, doi:10.1017/CBO9781139058452.

[28] Ratinov, L.; Roth, D.: Design Challenges and Misconceptions in Named Entity Recognition. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning*, CoNLL '09, Stroudsburg, PA, USA: Association for Computational Linguistics, 2009, ISBN 978-1-932432-29-9, s. 147–155.
URL http://dl.acm.org/citation.cfm?id=1596374.1596399

[29] Scikit image development team: Straight line Hough transform. web, 2017, [Online; navštíveno 23. 11. 2017].
URL http://scikit-image.org/docs/0.13.x/auto_examples/edges/plot_line_hough_transform.html

[30] Smith, R.: An Overview of the Tesseract OCR Engine. *Ninth International Conference on Document Analysis and Recognition (ICDAR 2007)*, ročník 2, 2007: s. 629–633, ISSN 1520-5363, doi:10.1109/ICDAR.2007.4376991.

[31] Smith, R. W.: *The extraction and recognition of text from multimedia document images*. Dizertační práce, University of Bristol, 11 1987.

[32] Zhou, G.; Su, J.: Named Entity Recognition Using an HMM-based Chunk Tagger. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, ACL '02, Stroudsburg, PA, USA: Association for Computational Linguistics, 2002, s. 473–480, doi:10.3115/1073083.1073163.
URL https://doi.org/10.3115/1073083.1073163

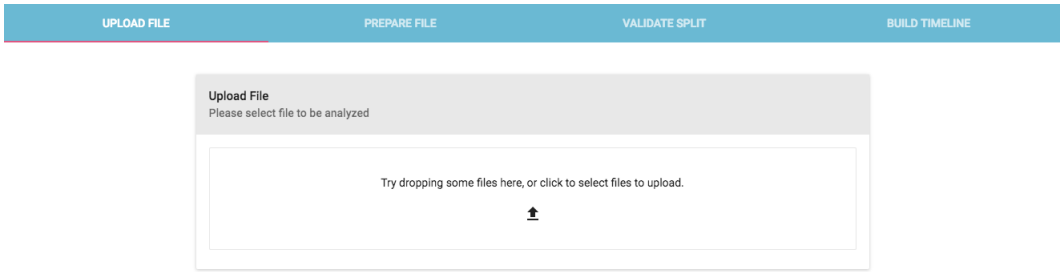
Přílohy

Seznam příloh

Příloha A

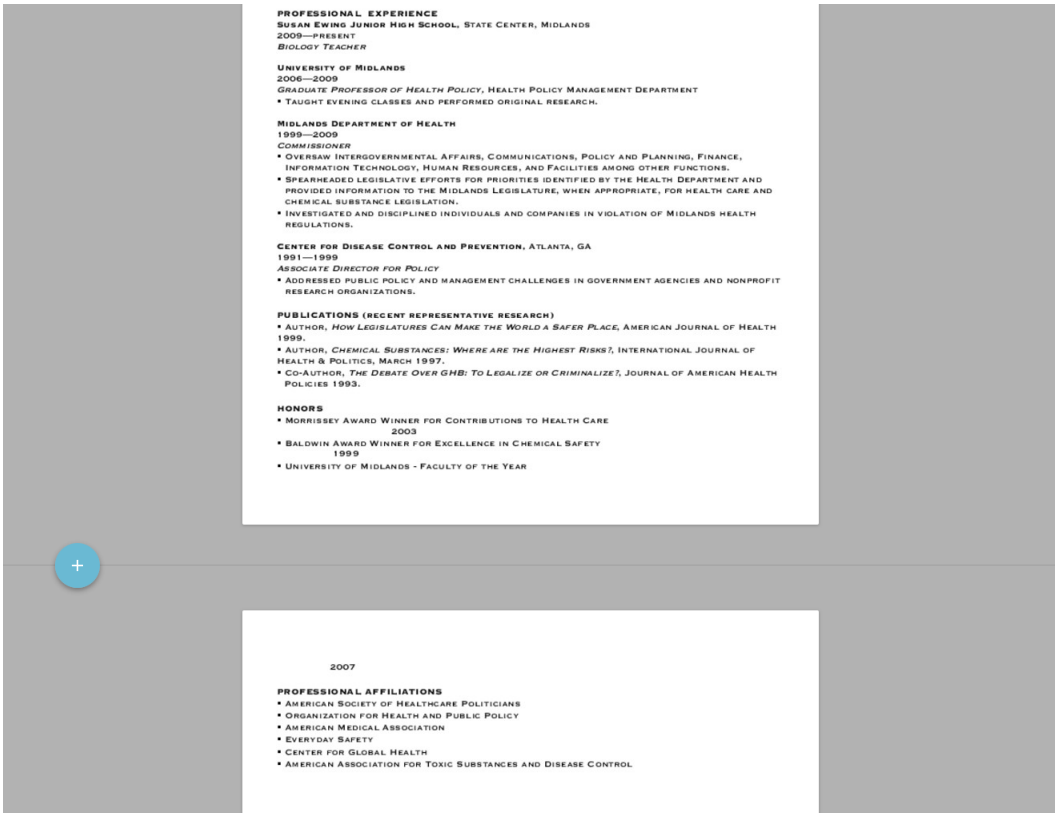
Implementovaná aplikace

A.1 Nahrání souboru



Obrázek A.1: Nahrání souboru [snímek obrazovky]

A.2 Detekce navazujícího dokumentu s možností editace



PROFESSIONAL EXPERIENCE
SUSAN EWING JUNIOR HIGH SCHOOL, STATE CENTER, MIDLANDS
2009—PRESENT
BIOLOGY TEACHER

UNIVERSITY OF MIDLANDS
2006—2009
GRADUATE PROFESSOR OF HEALTH POLICY, HEALTH POLICY MANAGEMENT DEPARTMENT
- TAUGHT EVENING CLASSES AND PERFORMED ORIGINAL RESEARCH.

MIDLANDS DEPARTMENT OF HEALTH
1999—2009
COMMISSIONER
- OVERSAW INTERGOVERNMENTAL AFFAIRS, COMMUNICATIONS, POLICY AND PLANNING, FINANCE, INFORMATION TECHNOLOGY, HUMAN RESOURCES, AND FACILITIES AMONG OTHER FUNCTIONS.
- SPEARHEADED LEGISLATIVE EFFORTS FOR PRIORITIES IDENTIFIED BY THE HEALTH DEPARTMENT AND PROVIDED INFORMATION TO THE MIDLANDS LEGISLATURE, WHEN APPROPRIATE, FOR HEALTH CARE AND CHEMICAL SUBSTANCE LEGISLATION.
- INVESTIGATED AND DISCIPLINED INDIVIDUALS AND COMPANIES IN VIOLATION OF MIDLANDS HEALTH REGULATIONS.

CENTER FOR DISEASE CONTROL AND PREVENTION, ATLANTA, GA
1991—1999
ASSOCIATE DIRECTOR FOR POLICY
- ADDRESSED PUBLIC POLICY AND MANAGEMENT CHALLENGES IN GOVERNMENT AGENCIES AND NONPROFIT RESEARCH ORGANIZATIONS.

PUBLICATIONS (RECENT REPRESENTATIVE RESEARCH)
- AUTHOR, *HOW LEGISLATURES CAN MAKE THE WORLD A SAFER PLACE*, AMERICAN JOURNAL OF HEALTH 1999.
- AUTHOR, *CHEMICAL SUBSTANCES: WHERE ARE THE HIGHEST RISKS?*, INTERNATIONAL JOURNAL OF HEALTH & POLITICS, MARCH 1997.
- CO-AUTHOR, *THE DEBATE OVER GHB: TO LEGALIZE OR CRIMINALIZE?*, JOURNAL OF AMERICAN HEALTH POLICIES 1993.

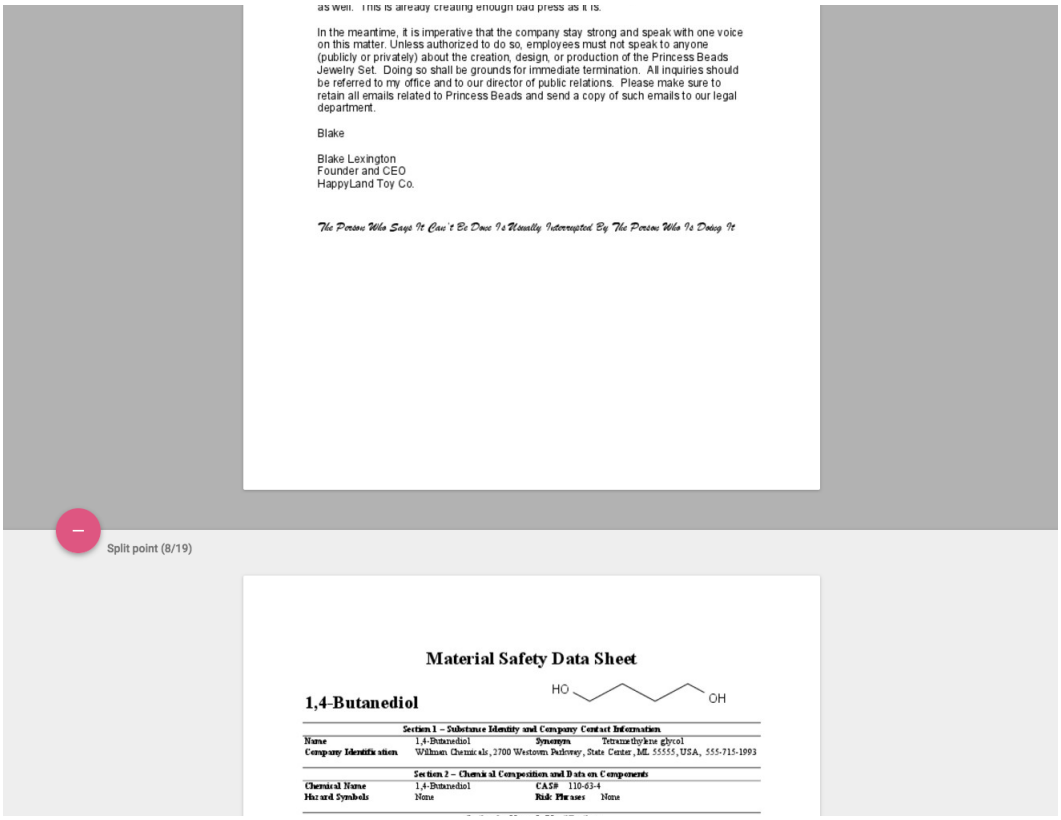
HONORS
- MORRISSEY AWARD WINNER FOR CONTRIBUTIONS TO HEALTH CARE
  2003
- BALDWIN AWARD WINNER FOR EXCELLENCE IN CHEMICAL SAFETY
  1999
- UNIVERSITY OF MIDLANDS - FACULTY OF THE YEAR

2007

PROFESSIONAL AFFILIATIONS
- AMERICAN SOCIETY OF HEALTHCARE POLITICIANS
- ORGANIZATION FOR HEALTH AND PUBLIC POLICY
- AMERICAN MEDICAL ASSOCIATION
- EVERYDAY SAFETY
- CENTER FOR GLOBAL HEALTH
- AMERICAN ASSOCIATION FOR TOXIC SUBSTANCES AND DISEASE CONTROL

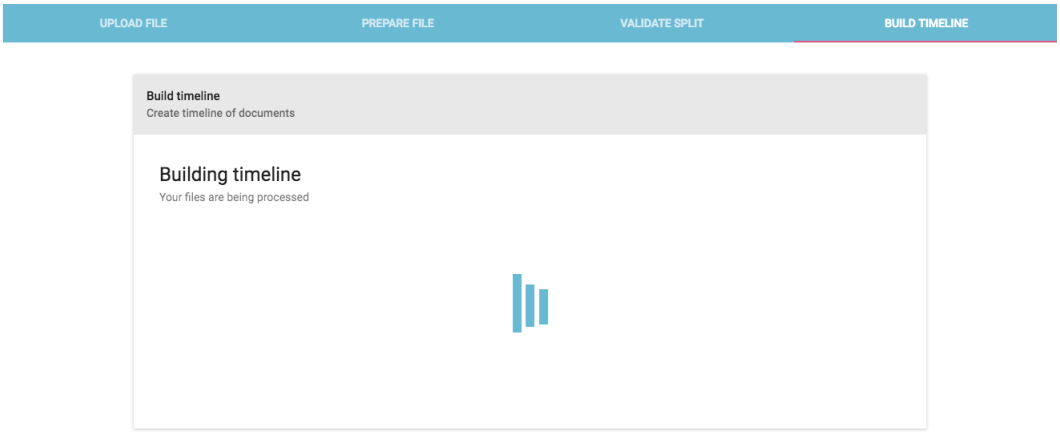
Obrázek A.2: Detekce navazujícího dokumentu s možností editace [snímek obrazovky]

A.3 Detekce dělícího bodu s možností editace



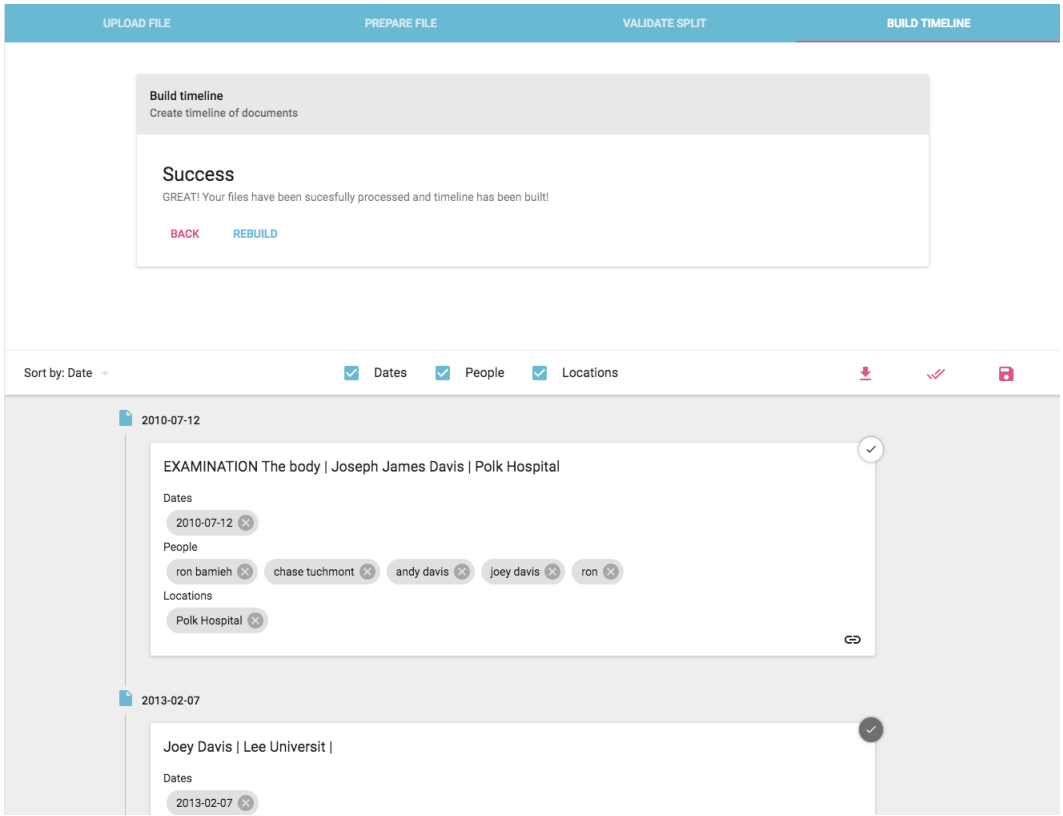
Obrázek A.3: Detekce dělícího bodu s možností editace [snímek obrazovky]

A.4 Tvorba časové osy



Obrázek A.4: Tvorba časové osy [snímek obrazovky]

A.5 Časová osa s možností editace



Obrázek A.5: Časová osa s možností editace [snímek obrazovky]

Příloha B

Obsah přiloženého paměťového média

Přiložené paměťové médium obsahuje:

- zdrojové kódy
- soubor readme.txt popisující obsah CD a instalační manuál
- skript install.sh pro instalaci potřebných závislostí
- tento dokument a dokument, který byl nahrán do IS
- testovací soubory
- snímky implementované aplikace
- videoukázka práce s aplikací