ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Název:	Návrh algoritmů pro podporu tvorby posudků
Student:	Jakub Vašíček
Vedoucí:	doc. RNDr. Ing. Marcel Jiřina, Ph.D.
Studijní program:	Informatika
Studijní obor:	Teoretická informatika
Katedra:	Katedra teoretické informatiky
Platnost zadání:	Do konce zimního semestru 2018/19

Pokyny pro vypracování

Cílem práce je navrhnout a ověřit algoritmy, které budou napomáhat uživateli vybrat vhodné obraty, které by uživatel mohl využít při psaní posudku. Cílem není napsat posudek za uživatele ani hodnotit práci, ale navrhovat („našeptávat“) uživateli vhodné obraty a znění textu na základě vstupních hodnotících údajů od uživatele. Hodnocení provádí uživatel, algoritmus pouze navrhuje možnosti vyjádření těchto hodnocení, které může uživatel dále upravovat.
1) Prozkoumejte existující řešení pro podporu tvorby posudků. Seznamte se s větnými formulacemi, které se vyskytují v různých druzích posudků kvalifikačních prací.
2) Navrhněte algoritmy, jak na základě vstupního hodnocení od uživatele (vyjádřeného bodovou škálou, výběrem klíčových slov apod.), kontextu místa a požadavků na posudek identifikovat vhodné formulace, které budou korespondovat se záměrem posuzovatele.
3) Navržené algoritmy implementujte a ověřte jejich funkčnost.
4) Proveďte praktické experimenty a zhodnoťte dosažené výsledky.

Seznam odborné literatury

Dodá vedoucí práce.

doc. Ing. Jan Janoušek, Ph.D.
vedoucí katedry

prof. Ing. Pavel Tvrdík, CSc.
děkan

V Praze dne 28. února 2017

České vysoké učení technické v Praze
Fakulta informačních technologií
Katedra teoretické informatiky

Bakalářská práce

Návrh algoritmů pro podporu tvorby posudků

Jakub Vašíček

Vedoucí práce: doc. RNDr. Ing. Marcel Jiřina, Ph.D.

15. května 2017

Poděkování

Děkuji svému vedoucímu doc. RNDr. Ing. Marcelu Jiřinovi, Ph.D. za užitečné rady a připomínky při vedení práce. Děkuji také své rodině za podporu po celou dobu studia.

Prohlášení

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

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona, ve znění pozdějších předpisů. V souladu s ust. § 46 odst. 6 tohoto zákona tímto uděluji nevýhradní oprávnění (licenci) k užití této mojí práce, a to včetně všech počítačových programů, jež jsou její součástí či přílohou a veškeré jejich dokumentace (dále souhrnně jen „Dílo“), a to všem osobám, které si přejí Dílo užít. Tyto osoby jsou oprávněny Dílo užít jakýmkoli způsobem, který nesnižuje hodnotu Díla a za jakýmkoli účelem (včetně užití k výdělečným účelům). Toto oprávnění je časově, teritoriálně i množstevně neomezené.

V Praze dne 15. května 2017 .....

Odkaz na tuto práci

Vašíček, Jakub. *Návrh algoritmů pro podporu tvorby posudků*. Bakalářská práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2017. Dostupný také z WWW: ⟨`http://vasicek-reviews-test.herokuapp.com/`⟩.

Abstrakt

Cílem této práce je navrhnout, implementovat a ověřit algoritmy, které v krocích napomohou hodnotiteli sestavit posudek na závěrečnou práci. Podpora spočívá zejména v nabízení vhodných formulací, které může uživatel začlenit do svého textu. Vhodné formulace k nabídnutí jsou identifikovány na základě hodnocení zadaného uživatelem. Nabídnuté formulace jsou seskupeny podle významu a podobné formulace je možno navzájem zaměňovat. Algoritmy pak pro ověření funkčnosti implementuji v podobě webové aplikace. Přínosem této práce je zejména zjednodušení práce hodnotitelů. Posudky často obsahují opakující se fráze a schémata, která aplikace navrhne sama a uživatel se pak může věnovat pouze samotnému hodnocení.

Klíčová slova posudky, kvalifikační práce, doporučování textu, kNN, hierarchické shlukování, interaktivita

Abstract

The goal of this thesis is to design, implement and test algorithms which will assist the user in writing a review of a final thesis. The assistance consists mostly of recommending formulations that the user can integrate into their text. Relevant formulations are identified according to the evaluation given by the user. The formulations are also split into groups by their meaning so that it is possible to interchange formulations with similar meanings. I implement the algorithms in the form of a web application. This work can be of use to reviewers and supervisors of student's theses — the reviews often contain repeating formulations which are suggested by the application and the user can concentrate solely on evaluating the thesis.

Keywords reviews, final theses, text recommendation, kNN, hierarchical clustering, interactivity

Obsah

Seznam obrázků

Seznam tabulek

Úvod

Prakticky každá lidská činnost poskytuje prostor, kde jí může napomoci stroj. Takové řešení nemusí vždy danou činnost zcela převzít, naopak ji lze pouze podporovat. Proto mohou existovat i nástroje, které usnadní činnost nesporně lidskou, jakou je psaní posudků kvalifikačních prací. Tvorba takového posudku zcela jistě vyžaduje lidskou subjektivitu a intuici, některé aspekty však lze usnadnit. Každá kvalifikační práce má mít určité kvality, které je potřeba ohodnotit. V posudku se pak často opakují formulace, které se k hodnocení těchto aspektů váží. Vezmeme-li v potaz také fakt, že požadavky na posuzování prací přichází nárazově vždy koncem semestru a počet vysokoškolských studentů neustále roste, naskýtá se možnost vytvořit nástroj, který by hodnotiteli ušetřil čas strávený vším schematickým, a poskytl více prostoru na samotné subjektivní ohodnocení dané kvalifikační práce.

Pokud je hodnotitel příliš zatížen zdánlivě zbytečnými formalitami, může na psaní posudku nahlížet jako na ztrátu svého času, což se pak může na hodnocení nebo kvalitě posudku projevit [1]. Výsledný nástroj by tedy měl být přehledný a práce s ním pohodlná a intuitivní. V opačném případě by se mohlo stát, že by nepříjemnosti se strukturou posudku pouze nahradily obtíže s nepohodlným používáním nové a neznámé aplikace. Těžištěm mé práce je sice návrh algoritmů, nicméně i zde je potřeba zohlednit jejich budoucí použití. Algoritmy by měly být navrženy tak, aby při implementaci dovolily aplikaci zaručit žádanou interaktivitu, měly by tedy být připraveny dynamicky reagovat na vstup uživatele.

Při výběru tématu pro mě byl důležitý také jeho „lidský aspekt“. Nejedná se o věc čistě technickou, ale je nutné analyzovat danou lidskou činnost a zamyslet se, jakým způsobem by bylo možné ji podpořit. Ve své práci tedy analyzuji strukturu posudku kvalifikační práce a následně navrhuji algoritmy, které na základě hodnocení zadaného uživatelem doporučují formulace, které by uživatel ve svém posudku mohl využít. Dále pak tyto algoritmy implementuji v podobě webové aplikace, která má sloužit zejména k ověření funkčnosti a umožnit analýzu navržených algoritmů.

Kapitola 1

Struktura posudku a existující řešení

V této kapitole popíši strukturu posudku, který bude pomocí navrhovaných algoritmů možno sestavit. Vysvětlím také výběr vzorových posudků a způsob tvorby databáze formulací, které pak mohu pomocí algoritmů navrhovat k doplnění do textu. Nakonec také okomentuji současný stav řešení této problematiky.

1.1 Výběr vzorových posudků

Jako první krok v mé práci bylo nutné sestavit si vhodnou sadu posudků, která může posloužit jako vzor při jejich následné analýze. Jelikož se má práce týká zejména posudků diplomových a bakalářských prací, jako nejlepší zdroj vzorů se naskýtá informační systém Fakulty informačních technologií ČVUT. V databázi se k datu 29. 4. 2017 nachází celkem 1860 závěrečných prací a k většině z nich je přiřazen posudek vedoucího práce i oponenta. Přestože posudek sestavený za pomoci navržených algoritmů nemá mít přesnou strukturu, jakou mají posudky závěrečných prací na FIT, je tato databáze natolik obsáhlá, aby mohla sloužit jako postačující zdroj vzorových textů.

Jako jazyk, se kterým budu primárně pracovat, jsem zvolil angličtinu. Oproti češtině má výhodu v jednodušší větné stavbě, slova se méně často ohýbají a lze se jednoduše vyhnout použití konkrétního rodu. Doporučené formulace tedy budou potřebovat méně editace před smysluplným začleněním do textu. S tímto omezením se počet posudků, které mohu použít jako vzor, značně snižuje, přesto by pro ukázku jejich počet měl zůstat dostatečný.

Při vybírání vzorových posudků jsem si však všiml, že naprostá většina posudků psaných v angličtině obsahuje pozitivní hodnocení. Vysvětluji si to skutečností, že píše-li student svou závěrečnou práci v angličtině, jedná se často o výsledek spolupráce se zahraniční firmou nebo o výstup ze zahraniční stáže.

Takovéto práce většinou nebudou odbyté a jejich kvalita je nadprůměrná. Negativních posudků je v databázi menšina a anglických se mi nepodařilo najít dostatečné množství, musel jsem tedy mezi vzorovou sadu začlenit i posudky psané česky, abych pokryl celou škálu možných hodnocení. Vybral jsem takto 40 vzorových posudků, ze kterých jsem pak sestavil vzorovou databázi použitelných formulací, věty z českých posudků jsem pak před začleněním do databáze přeložil do angličtiny sám.

Při výběru formulací, které by mohly být v našeptávači použity, jsem se soustředil hlavně na věty, které jsou nějakým způsobem univerzální, a dají se po snadném doplnění použít. Nemusí to však nutně znamenat, že daná věta bude bezobsažná — v závěrečných pracích se často posuzují stejné kvality, používané formulace se tedy budou opakovat, pouze budou komentovat různé jevy. Konkrétní názvy nebo pojmy tedy mohu z věty vyjmout a nahradit je symbolem, který značí, že by na tomto místě měla být věta doplněna. Například z věty *„Although the topic of the thesis is related to problems of a future traffic infrastructure, its difficulty throughout the text does not pose an extremely challenging research task.“* může vzniknout vzor *„Although the topic of the thesis is related to ___, its difficulty does not pose an extremely challenging task.“*, který může dobře vystihnout určité hodnocení libovolného tématu. Vyberu–li takovýchto vzorů dostatečné množství pro vyjádření každého hodnocení, nebudou se nutně v každém posudku opakovat ty stejné. Pokud by nadále výběr formulací hodnotitele neuspokojil, bude moct jakoukoli libovolně upravit, či nepoužít žádnou z nich.

1.2 Struktura posudku a hodnocené aspekty práce

Požadovaná struktura posudků, které budou vytvářeny za podpory navrhovaných algoritmů, není tak přesně daná, jak je to u posudku na FIT. Ten se standardně dělí na 11 částí, přičemž vedoucí práce vynechává část *Otázky k obhajobě* a oponent část *Aktivita a samostatnost studenta v průběhu řešení.* Výpis sekcí předkládám v tabulce 1.1.

Mým záměrem je, aby byly navrhované algoritmy využitelné i mimo prostředí fakulty. Budu tedy předpokládat, že vytvářený posudek bude spíše souvislý text, který bude obsahovat odstavce místo oddělených komentářů k jednotlivým hodnoceným aspektům. Bude–li však žádoucí použít jinou strukturu, nemělo by být složité návrh příslušným způsobem upravit. Ve svém návrhu jsem tedy sekce z tabulky 1.1 sloučil do 7 odstavců následujícím způsobem:

- **Originality and difficulty of the topic**: Zde se specifikuje, jakého tématu se hodnocená práce týká. Hodnotí se pak náročnost a originalita zadání.
- **Factual level of the thesis and bibliography**: Zde se hodnotí studentovy teoretické znalosti, přehled o problematice a práce s literaturou.

- **Formal level of the written thesis**: Tento odstavec se týká formální kvality textu. Hodnotí se struktura, logické rozčlenění do kapitol, konzistence v užívání termínů, dále pak úroveň jazyka a typografická kvalita textu.
- **Fulfilment of the assignment and applicability of the results**: V tomto odstavci se hodnotí výsledek práce: zda splňuje zadání a jak je v praxi využitelný.
- **Activity and self-reliance of the student**: Zde vedoucí práce hodnotí studentovu aktivitu a schopnost komunikovat a prezentovat průběžné výsledky. Tento odstavec musí zůstat samostatně, protože tato témata oponent práce pro svůj posudek nevyužije.
- **Questions for the defence**: Otázky k obhajobě — tato část bude naopak využívána spíše oponentem.
- **Overall evaluation**: Zde je prostor pro shrnutí všech hodnocených aspektů a návrh výsledné známky.

Tabulka 1.1: Struktura posudku na FIT

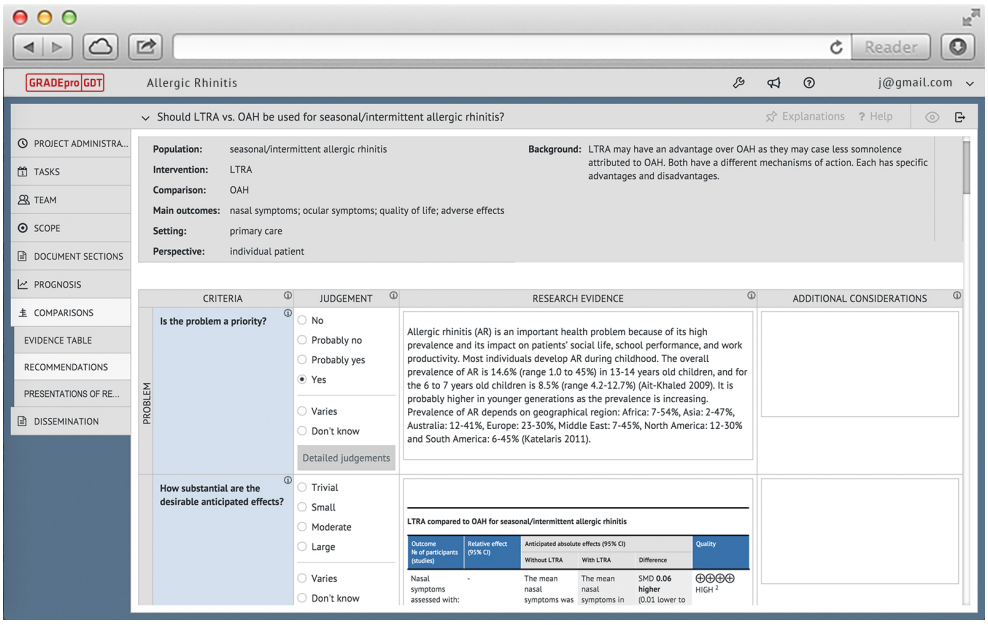
Anglická verze	Česká verze
Difficulty and other comments on the assignment	Náročnost a další komentář k zadání
Fulfilment of the assignment	Splnění zadání
Size of the main written part	Rozsah písemné zprávy
Factual and logical level of the thesis	Věcná a logická úroveň práce
Formal level of the thesis	Formální úroveň práce
Bibliography	Práce se zdroji
Evaluation of results, publication outputs and awards	Hodnocení výsledků, publikační výstupy a ocenění
Applicability of the results	Komentář o využitelnosti výsledků
Activity and self-reliance of the student	Aktivita a samostatnost studenta v průběhu řešení
Questions for the defence	Otázky k obhajobě
The overall evaluation	Celkové hodnocení

Takto některé odstavce mohou obsáhnout více aspektů práce, které spolu souvisí, a vytvořit souvislejší text. Navržená struktura stále dodržuje určitý

průběh posudku, kdy se nejprve okomentuje téma nebo zadání práce, následně určitá formální a faktická specifika, dále se hodnotí výstup, který práce poskytuje, následně přístup studenta (pokud je možnost toto hodnotit). Posudek by měl vždy končit shrnutím všeho předchozího a návrhem výsledného hodnocení či známky.

1.3 Existující řešení

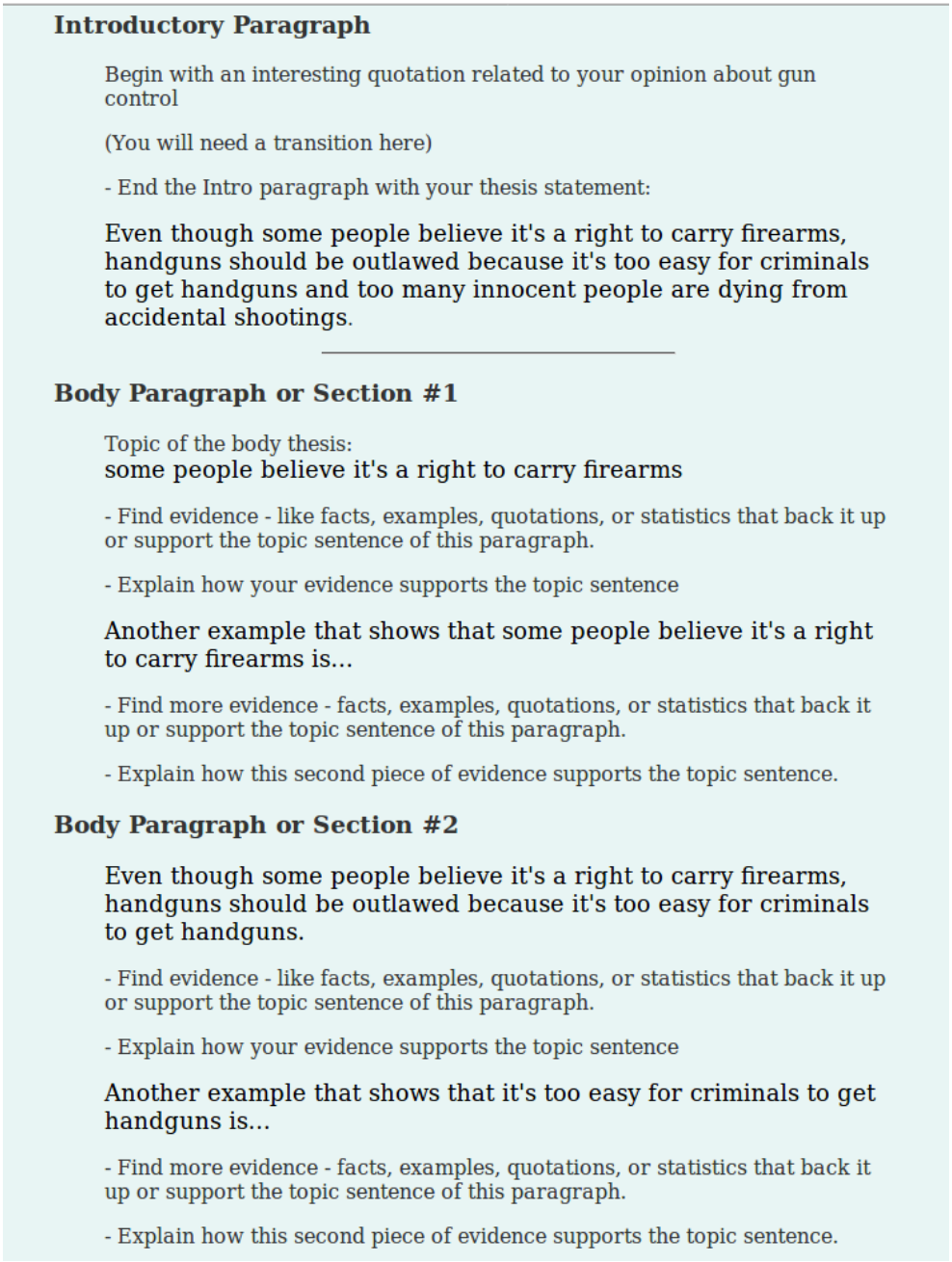
Jedná se o poměrně specifické zadání, ke kterému příliš mnoho referencí nebylo možné dohledat. Nepodařilo se mi tedy nalézt žádný nástroj, který by poskytoval podobnou funkcionalitu pro psaní posudků. Pod hesly jako „review writing tool“ jsou na Google k nalezení různé návody, jak provést kvalitní rešerši ke kvalifikační práci či vybrat vhodné zdroje. Nalezl jsem článek, kde se ukazuje použití aplikace Gradepro GDT pro sepsání přehledu literatury [2]. Popisovaná práce s aplikací však působí velmi komplikovaně a nijak se nepodobá představě, kterou o své práci mám. Tato aplikace zřejmě slouží také k vyhledávání článků a je zaměřena na oblast zdravotnictví [3]. Na webu Openingscience.org jsem dále našel odkaz na aplikaci CiteULike, kde zmiňují funkci „review tool“, domnívám se však, že se opět jedná o sestavování přehledu literatury, neboť při používání této aplikace jsem žádnou funkci psaní posudku nenašel [4].



Obrázek 1.1: Uživatelské rozhraní aplikace Gradepro GDT [3]

Na zajímavější výsledek jsem narazil v myšlenkové mapě, kterou vytvořil Nader Ale Ebrahim jako přehled dostupných nástrojů, které lze využít při psaní článku [5]. Nachází se zde v uzlu „Writing a paper“ skupina označená jako „Thesis Generators“. Jedná se o skupinu podobných nástrojů, které na základě myšlenek, které uživatel formuluje v podobě hesel nebo frází, dokáží rozvrhnout strukturu argumentu předkládaného v eseji. Jako příklad mohu uvést nástroj Thesis Builder [6]. Po vyplnění požadovaných formulací navrhne strukturu eseje a ke každému odstavci sestaví ukázkovou větu, která by měla shrnovat požadované sdělení. Společné s mým návrhem je zde rozdělení do několika standardních odstavců, které mají definované, jaké sdělení by měly nést. Na rozdíl od mého návrhu se zde však vyžaduje vyplnění chybějících částí textu dříve, než je výsledný text sestaven. Já jsem zvolil přístup opačný, tedy nejprve navrhnout uživateli strukturu textu a vzory použitelných formulací, které si uživatel následně doplní a upraví.

Obrázek 1.2: Ukázka práce s aplikací Thesis Builder [6]



Obrázek 1.3: Ukázka výstupu aplikace Thesis Builder [6]

Kapitola 2

Návrh řešení

V této kapitole se věnuji přesnému návrhu algoritmů, které budou podporovat sestavení posudku. Proces sestavení posudku rozděluji do pěti fází. Dále diskutuji automatické doplňování klíčových slov do nabídnutých vět a navrhuji způsob identifikace vzájemně ekvivalentních formulací. Nakonec popisuji způsob identifikace formulací vhodných k doporučení, průběh doporučování a reakce na vstup uživatele.

Jak už výběr vzorových posudků, vět a struktury nabízí, budu uvažovat použití navrhovaných algoritmů zejména pro posudky prací v informatických či spřízněných technických oborech. V jiných oborech by například závěrečné práce nemusely zahrnovat praktický výstup a zcela jistě nebudou využívat specifické technologie. Pro sestavování posudků prací v jiných oborech by bylo potřeba sestavit jinou databázi vhodných vět, změnit dotazy ke klíčovým slovům a v případě nutnosti pozměnit navrženou strukturu posudku. Algoritmy týkající se doporučování formulací by však měly být použitelné univerzálně.

2.1 Proces tvorby posudku

Aby mohl vzniknout nástroj, který provede uživatele sestavením posudku od začátku do konce, je potřeba rozdělit celou tvorbu posudku do několika částí. Podle tohoto rozvržení pak budu moct navrhnout algoritmy, které budou sestavení posudku podporovat.

Obrázek 2.1: Postup při sestavení posudku

1. Úvod do aplikace, specifikace druhu práce (bakalářská, diplomová, ...). Zde by se měl uživatel zorientovat v prostředí a vyplnit několik základních údajů o posuzované práci, z hlediska návrhu algoritmů není tento krok příliš zajímavý.

2. V tomto místě by měl uživatel potvrdit, zda chce v práci hodnotit všechny nabízené aspekty, nebo zda chce některé z nich vynechat. Také by bylo vhodné zde uvést, jakého tématu se hodnocená práce týká, ideálně pomocí několika klíčových slov nebo hesel.

3. Zde je struktura posudku připravena a uživatel může přejít k ohodnocení práce. Dotazy kladené uživateli by měly být odpovídající větám, které lze následně doporučit. V tomto kroku se tedy bude nacházet hlavní kritérium, podle kterého budou identifikovány věty vhodné k doplnění do textu.

4. V tomto kroku je vše připraveno pro samotné sestavení posudku. Zde by se měl v aplikaci nacházet „chytrý“ textový editor, který by vytvořil předem definovanou strukturu posudku, a dále by obsahoval našeptávač, který by uživateli nabízel co nejvhodnější formulace k doplnění do textu. Zde doporučované formulace by měly být seřazeny primárně podle hodnocení zadaného v předchozím kroku, ale našeptávač by měl být schopný reagovat na aktivitu uživatele a „vhodnost“ doporučovaných formulací v případě potřeby přehodnotit.

5. Nakonec by mělo být uživateli umožněno přečíst si posudek jako souvislý text, udělat finální úpravy a nakonec jej uložit ve vhodném formátu.

Jak je patrné, z hlediska algoritmů pro doporučování vhodných vět budou nejzajímavější kroky 2–4. Prvnímu a poslednímu kroku se dále v návrhu již věnovat nebudu, ty budou hrát roli až v implementaci, aby ukázková aplikace byla kompletně použitelná pro sestavení posudku.

2.2 Klíčová slova

Klíčová slova zadaná v druhém kroku jsem původně zamýšlel automaticky doplňovat na příslušná místa ve větách. Po experimentu jsem však tuto myšlenku zavrhl ze dvou důvodů. Bohužel i v angličtině je někdy potřeba slova ohýbat a doplnit automaticky do věty chybějící výraz tak, aby věta nepotřebovala další editaci, by byl úkol velmi náročný. Navíc bych takto mohl nakonec pouze limitovat možnosti uživatele, jak danou formulaci použít. Proto tato klíčová slova nabídnu uživateli mimo doporučené věty tak, aby je mohl na chybějící místa jednoduše doplnit, např. jedním kliknutím.

Abych uživateli napověděl, která specifika má klíčovými slovy popsat, stanovil jsem čtyři otázky, na které mají klíčová slova odpovídat:

1. Co bylo náplní autorovy práce? (např. analýza, návrh, implementace)
2. Jaké problematice se autor věnuje? (např. jaderná fyzika, rozpoznávání hlasu, predikátová logika)
3. Jaké technologie autor používá? (např. jQuery, Java Spark, MySQL)
4. Co je výsledkem autorovy práce? (např. aplikace, algoritmus, návrh)

Takto by zadaná klíčová slova mohla pokrývat velkou část výrazů, které bude následně při sestavování posudku potřeba doplňovat do doporučených formulací.

2.3 Hodnocení posuzované práce

Budu předpokládat strukturu posudku, jak jsem ji definoval v předchozí kapitole. Každému ze sedmi odstavců jsem přiřadil sadu dotazů, které by měly pokrýt všechny aspekty práce, které jsou v daném odstavci hodnoceny. Tyto otázky budou uživateli předloženy v třetím kroku při hodnocení práce, odpovědí by vždy mělo být číslo v rozmezí 0–100. Definoval jsem si tři různé druhy otázek:

- Ano/Ne — Zde budou možné pouze tyto dvě odpovědi, hodnocení tedy bude vždy přesně 100 (Ano) nebo 0 (Ne). Tento druh hodnocení jsem nakonec k žádné otázce nevyužil, protože výběr odpovědí je příliš omezený. Nechávám jej zde ale jako možnost, se kterou jsem původně počítal, pouze nakonec nebyla využitá.
- 4 stupně — Druhou možností je rozhodnutí, do jaké míry uživatel souhlasí s určitým tvrzením. Výsledkem jsou čtyři možná hodnocení: 100 (Souhlasím), 66 (Souhlasím s menšími výhradami), 33 (Souhlasím s velkými výhradami) a 0 (Nesouhlasím). Zde jsem se inspiroval hodnocením, které se používá u posudků na FIT a používám je, když se jedná o velmi konkrétní otázku, resp. tvrzení. V tomto případě je pro uživatele jednodušší rozhodnout se, zda toto tvrzení platí zcela, zčásti nebo vůbec, než jeho platnost hodnotit na celé bodové škále.
- 0–100 — Poslední možností je nechat uživatele vybírat hodnocení na celé škále od 0 do 100. U otázky, která je obecnější, tak nechávám uživateli větší volnost v co nejpřesnějším ohodnocení dané kvality práce. Tato škála by měla být také dobře srozumitelná, protože je ekvivalentní procentuálnímu hodnocení.

Otázky k jednotlivým odstavcům volím tak, aby co nejlépe vystihovaly jejich předpokládaný obsah, který jsem popsal v předchozí kapitole.

1. Originality and difficulty of the topic
    - Téma této práce je aktuální nebo inovativní. (4 stupně)
    - Jak náročné je toto zadání? (0–100)
2. Factual level of the thesis and bibliography
    - Rozsah práce splňuje požadavky. (4 stupně)
    - Všechny části obsahují pouze potřebné informace. (4 stupně)
    - Má autor o tématu dobrý přehled? (0–100)
    - Používá autor relevantní zdroje informací? (0–100)
    - Autor cituje všechny použité zdroje. (4 stupně)
3. Formal level of the written thesis
    - Je práce logicky rozčleněná do kapitol a dobře čitelná? (0–100)
    - Autor nepoužívá nejasné formulace. (4 stupně)
    - Dosahuje jazyk použitý v práci předpokládané úrovně? (0–100)
    - Práce neobsahuje překlepy ani typografické chyby. (4 stupně)
4. Fulfilment of the assignment and applicability of the results
    - Splnil autor svůj cíl? (0–100)
    - Výsledky jsou validní a dobře podložené. (4 stupně)
    - Výsledky mohou být použity v praxi nebo publikovány. (4 stupně)
5. Activity and self-reliance of the student
    - Pracoval student aktivně a samostatně? (0–100)
    - Projevoval student zájem o dané téma? (0–100)
    - Student komunikoval s vedoucím bez problému po celou dobu. (4 stupně)
    - Student vždy dodržel stanovený termín. (4 stupně)
6. Questions for the defence
    - Zde je prostor pro konkrétní dotazy, doporučovat často používané formulace v tomto případě nemá smysl, proto zde nebude ani žádná otázka.
7. Overall evaluation
    - Ohodnoťte známkou celou práci. (0–100)

Může se zdát, že první otázka u druhého odstavce by měla patřit spíše k odstavci třetímu. Hodnocení rozsahu práce se však většinou odráží i od jejího obsahu, protože například i kratší, ale obsahově kvalitní práce může být kladně hodnocená. Proto se tyto dva aspekty komentují v posudcích společně a hodnocení rozsahu práce tedy řadím k druhému odstavci.

2.4 Identifikace vhodných formulací

Identifikace vhodných formulací k doporučení se dá chápat jako určitá forma klasifikace. Uložené věty v databázi si mohu označit podle hodnocení, k jakému se váží, a pak je promítnout do n-rozměrného prostoru, kde n je počet otázek náležících k danému odstavci. Odpovědi na tyto otázky jsou vždy hodnoty normalizované do intervalu $[0, 100]$, proto se dá odpovídajícím způsobem vypočítat vzdálenost věty od aktuálního hodnocení.

V tomto prostoru je pak dobře využitelný klasifikační algoritmus **kNN** (k nejbližších sousedů), pomocí kterého mohu nalézt stanovený počet nejvhodnějších formulací. Tento algoritmus je dobře využitelný u dat, jejich rozdělení není předem známo [7]. Tato vlastnost je velmi užitečná, protože je důležité, aby mnou navržené algoritmy byly využitelné pro adekvátní výběr formulací z libovolné databáze bez ohledu na to, jak jsou věty rozmístěny v prostoru.

Rozdíl proti standardní podobě algoritmu je ten, že jde zejména o samotné nalezení nejbližších sousedů k aktuálnímu hodnocení, pro které se již pak žádná výsledná třída neurčuje.

2.4.1 Ohodnocení věty v databázi

Předpokládám, že se věty doporučované v jednotlivých odstavcích nebudou opakovat, proto mohu pro každý odstavec držet samostatnou databázi vět, ze kterých následně bude možné vybírat. Věty v databázi budou uloženy tímto způsobem:

Tabulka 2.1: Uložení věty v databázi

Cluster	Position	Attr1	Attr2	Attr3	Value
-1	1	100	-1	100	The ___ is functional and is very open to be extended in the future.
2	2	75	-1	-1	Therefore the final result cannot be excellent.
6	1	-1	-1	25	The results as they are now are not publishable.

Nyní objasním význam jednotlivých sloupců v tabulce 2.1:

- **Cluster** značí pořadí shluku podobných vět, ve kterém se daná věta nachází. Hodnota -1 značí, že věta do žádného shluku dosud zařazena nebyla. Shlukování vět s podobným významem se budu věnovat později.
- **Position** značí pozici věty v textu. Původně jsem zamýšlel, že se zde bude nacházet předpokládané pořadí věty v textu daného odstavce. Některé odstavce se však týkají více jevů najednou a není jasné, který z nich

bude uživatel chtít komentovat jako první. Jediné, co lze v tomto ohledu o větě jasně říct, je, zda věta dané téma uvozuje, nebo předpokládá, že o tomto jevu uživatel již něco napsal. Používám zde tedy pouze hodnoty 1 a 2.

- Atributy **Attr** obsahují předpokládané hodnocení práce podle otázek uvedených v předchozí sekci, při kterém by měla být věta použita. Jejich počet tedy musí být stejný jako počet otázek k danému odstavci. Nachází-li se v atributu hodnota -1, znamená to, že na odpovědi uživatele na danou otázku při výběru věty nezáleží. V uvedeném případě jsou věty vybrány z databáze k odstavci 4. Hodnoty atributů první věty říkají, že stanovený cíl byl zcela splněn a výsledek je bez problému použitelný nebo publikovatelný.

- Hodnota **Value** obsahuje samotnou větu k doporučení.

Při sestavování databáze k poslednímu odstavci, který má shrnout obsah všech předchozích, vznikla nutnost odkazovat se i na hodnocení jiných odstavců. Toto vyžaduje složitější strukturu, kde se musím odkazovat první na odstavec, pak na konkrétní otázku a až pak uvést předpokládané hodnocení. Kdybych v návrhu zaváděl přesnou podobu této úpravy a snažil se ji promítnout do vzorce pro výpočet vzdálenosti, pravděpodobně by se stal následující popis nepřehledným. Proto zde budu diskutovat důsledky této úpravy pouze slovně a konkrétní příklad uvedu až v následující kapitole.

2.4.2 Identifikace vět s podobným významem

Jak jsem již zmínil, je nutné, aby databáze obsahovala věty, které jsou si významově podobné. Jedině takto bude moct uživatel vyjádřit svou myšlenku více různými způsoby bez toho, aby nabízené věty nevyužíval a celý text napsal sám. Potenciálních vět k doplnění do textu bude ve většině případů celá řada, bude tedy nutné nabízet je postupně. Zde by pak mohla nastat situace, kdy budou všechny momentálně nabízené věty vyjadřovat stejnou myšlenku. Tato situace je samozřejmě nežádoucí a bude potřeba jí předcházet. Řešením je rozdělit věty do skupin podle významu a z každé skupiny vždy uživateli nabídnout k doplnění „reprezentanta“, čili nejadekvátnější větu, kterou v případě potřeby bude moct vyměnit za jinou ze skupiny.

První možností by mohlo být nedržet v databázi samostatné věty, ale rovnou celé skupiny podobných vět. Takto složitě strukturovaná databáze by se však velmi nepohodlně udržovala. Při doplnění nové věty by první bylo nutné nalézt vhodnou skupinu a až do ní větu zařadit. Věty však jsou poměrně dobře popsány zmíněnými atributy, ze kterých lze jejich význam vyčíst. Proces rozřazení do skupin je tedy možné automatizovat.

Protože neexistuje žádný vzor, podle kterého by se daly věty zařadit do skupin, bude nejvhodnější použít metodu shlukování. Shlukovacích algoritmů

existuje velké množství, při výběru jsem vycházel zejména z knihy *Data Clustering: Theory, Algorithms and Applications* [8].

Skutečnost, že není předem známý počet shluků, do kterých mají být věty seskupeny, vyřazuje možnost použít nejběžnější algoritmus *k-means*. Nejvhodnější tedy bude použít některou z technik hierarchického shlukování, konkrétně shlukování aglomerativní. Uvažuji situaci, kdy je databáze nová a věty zatím nebyly seskupeny. V prvním kroku jsou všechny věty přiřazeny do samostatného shluku o jednom prvku. V každém dalším kroku se pak vypočte vzdálenost mezi každou dvojicí shluků a nejbližší dva se sloučí dohromady. Takto se shluky slučují, dokud minimální vzdálenost mezi dvěma shluky nedosáhne určité hranice, pak se shlukování zastaví. Výsledek pak závisí zejména na volbě metody výpočtu vzdálenosti a minimální požadované vzdálenosti shluků.

Když jsou jednou věty seskupeny, jejich příslušnost do shluků se již nebude měnit, je tedy žádoucí si výsledek shlukování uložit, aby se při každém dalším čtení databáze nemuselo provádět znovu. Pokud mezi dvěma přístupy v databázi přibude nezařazená věta, vytvoří se pro ni opět samostatný shluk a před následujícím čtením se spustí stejný algoritmus pro aglomerativní shlukování. Takto se nová věta zařadí na „své místo“ a výsledek se může opět uložit pro další čtení.

U hierarchického shlukování existuje ještě druhý přístup, kdy se v prvním kroku všechny instance spojí do jediného shluku, a ten se následně dělí na menší části. Použití tohoto přístupu by však nepodporovalo přidávání samostatných nových vět do již spočtené struktury shluků, proto jej nepoužívám.

2.4.3 Výpočet vzdálenosti dvou shluků vět

Aby bylo možné vypočítat vzdálenost dvou shluků, bude nejprve potřeba definovat vzdálenost dvou vět. Označím si počet atributů (tedy počet otázek k danému odstavci) jako n. Vzdálenost dvou vět x a y pak definuji standardně jako Eukleidovskou vzdálenost $d(x,y) = \left[\sum_{i=1}^{n}(x_i - y_i)^2\right]^{\frac{1}{2}}$ [8].

Musím však uvažovat případ, kdy i-tý atribut věty x je roven -1, tedy na něm při doporučování nezáleží, a zároveň i-tý atribut věty y roven -1 není, nebo naopak. V tomto případě věty x a y nemohou mít podobný význam a nemohou se nacházet ve stejném shluku. Definici vzdálenosti vět musím tedy rozšířit následovně:

$$d(x,y) = \begin{cases} \infty, & \exists(i \leq n) : (x_i = -1 \wedge y_i \neq -1) \vee (y_i = -1 \wedge x_i \neq -1), \\ \left[\sum_{i=1}^{n}(x_i - y_i)^2\right]^{\frac{1}{2}}, & jinak. \end{cases}$$

Stejná podmínka bude platit také, pokud se věty odkazují na různé otázky z jiných odstavců, v tomto případě se také nemohou nacházet ve stejném shluku a vzdálenost je potřeba nastavit na nejvyšší možnou. Pokud se dvě věty budou dokazovat na tytéž otázky z jiných odstavců, je samozřejmě třeba

použít ve výpočtu vzdálenosti i tyto hodnoty. Tuto podmínku nebudu zavádět formálně, ale je nutné zmínit, že s tímto musím při implementaci také počítat.

Metod, jak spočítat vzdálenost dvou shluků, opět existuje nemalé množství. Rozeberu zde použití tří z nich. Pro ukázku chování jednotlivých metod budu používat tento příklad: V databázi mám uloženy čtyři věty a, b, c a d s následujícími atributy: $a = [100, -1]$, $b = [90, -1]$, $c = [75, -1]$ a $d = [-1, 100]$. Jako minimální požadovanou vzdálenost dvou shluků si určím hodnotu 20.

- Metoda **single-link**: Vzdálenost mezi shluky C a C' se vypočítá pomocí vzorce $D(C, C') = min\ d(x, y);\ x \in C, y \in C'$ [8]. Vzdálenost mezi dvěma shluky je tedy definována jako vzdálenost mezi nejbližší dvojicí instancí. Takto vzniklé shluky bývají větší a snaží se pokrýt celou souvisle zaplněnou plochu. V ukázkovém případě by se v prvním kroku sloučily věty a a b. V druhém kroku by vzdálenost shluku $\{a, b\}$ od věty c byla 15, což stále splňuje podmínku, takže by vznikly dva shluky $\{a, b, c\}$ a $\{d\}$. To by však znamenalo, že v jednom shluku jsou spolu věty a a c, které mají vzájemnou vzdálenost 25 a mohou se již významem lišit. Proto chování této metody tedy není pro mé potřeby ideální.

- Metoda **complete-link**: $D(C, C') = max\ d(x, y);\ x \in C, y \in C'$ [8]. Vzdálenost mezi dvěma shluky se definuje jako vzdálenost mezi dvojicí navzájem nejvzdálenějších instancí. V ukázkovém případě by v druhém kroku ke sloučení věty c se shlukem $\{a, b\}$ nedošlo, protože vzdálenost mezi nimi by nyní byla 25. Takové chování je pro můj případ vhodnější, protože neumožňuje sloučit dva shluky, které obsahují významově odlišné věty.

- Metoda využívající **centroid**: Zde se definuje centroid shluku C $\mu(C)$ jako pomyslný „střed“ shluku. Jeho atributy se při inicializaci nastaví jako atributy jediné věty ve shluku, a při každém sloučení dvou shluků C a C' se nový centroid vypočte jako $\mu(C \cup C') = \frac{|C|\mu(C)+|C'|\mu(C')}{|C|+|C'|}$ [8]. Vzdálenost dvou shluků se pak rovná vzdálenosti jejich centroidů, tedy $D(C, C') = d(\mu(C), \mu(C'))$ [9]. Situace v druhém kroku ukázkového případu bude následující: shluk $\{a, b\}$ má centroid s atributy $[95, -1]$. Jeho vzdálenost od věty c bude 20 a ke sloučení tedy opět nedojde. Toto chování je podobné předchozí metodě a také se jeví jako vyhovující.

Jako použitelné se jeví metody *complete-link* a *centroid*. Nejdůležitější v mém případě je, aby žádný shluk neobsahoval dvě věty, které by se svým významem zásadněji lišily. Tyto situace nejlépe eliminuje *complete-link*, proto budu používat k výpočtu vzdálenosti mezi dvěma shluky tuto metodu.

Hodnota **Position** se ve vzájemné vzdálenosti neprojeví a bude mít význam až v průběhu doporučování vět uživateli.

2.4.4 Výpočet vzdálenosti shluku od aktuálního hodnocení

Nyní se dostávám k situaci, kdy jsou věty načteny z databáze, rozděleny do shluků a uživatel ohodnotil posuzovanou práci (tedy dokončil třetí krok tvorby posudku). Z důvodů zmíněných výše budu chtít doporučovat celé shluky vět. Abych je mohl seřadit od nejvhodnějších po nejméně odpovídající, potřebuji zavést vzdálenost shluku vět od aktuálního hodnocení. Pro výpočet této vzdálenosti budu opět potřebovat vzdálenost jednotlivé věty od hodnocení, kterou opět definuji jako Eukleidovskou vzdálenost s několika úpravami.

První úpravu vyžaduje skutečnost, že věta se netýká vždy všech hodnocených aspektů v daném odstavci, tedy některé její atributy ponesou hodnoto -1. Pro výpočet tedy budu používat pouze ty atributy, jejichž hodnota bude různá od -1 — jinými slovy z výpočtu Eukleidovské vzdálenosti musím některé dimenze vynechat. Formálně bych mohl definovat množinu I_x a dále vzdálenost $d(x, h)$, kde x je věta z databáze a h je vektor obsahující hodnoty aktuálního hodnocení k danému odstavci, tímto způsobem:

$$I_x = \{i | x_i \neq -1, i \leq n\},$$
$$d(x, h) = \left[\sum_{i \in I_x} (x_i - h_i)^2\right]^{\frac{1}{2}}.$$

Při výpočtu Eukleidovské vzdálenosti se však počítá s tím, že všechny porovnávané elementy mají stejný počet dimenzí. Tato podmínka je předchozí úpravou porušena a takto spočítané vzdálenosti již nebudou ve všech případech odpovídající. Například pro hodnocení $h = [85, 90, 95]$ a věty $x = [95, -1, -1]$ a $y = [95, -1, 85]$ vychází vzdálenosti $d(x, h) = 10$ a $d(y, h) \doteq 14,142$. Je vidět, že věty s více dimenzemi jsou „znevýhodněny“. Jinými slovy, věta lišící se o 10 jednotek v jedné dimenzi je bližší než věta lišící se o 10 jednotek ve dvou dimenzích. Věty týkající se více témat najednou by takto byly doporučovány s menší pravděpodobností.

Abych se této situaci vyhnul, bude nutné výpočet vzdálenosti „normalizovat“ na libovolný počet dimenzí. K tomu slouží jednoduchá úprava Eukleidovské vzdálenosti na *průměrnou vzdálenost*: $d_{ave}(x, y) = \left[\frac{1}{n}\sum_{i=1}^{n}(x_i - y_i)^2\right]^{\frac{1}{2}}$ [8]. Před odmocněním tedy navíc součet vydělím počtem dimenzí, ve kterých se počítá rozdíl souřadnic. Výsledný vzorec pro výpočet vzdálenosti věty od aktuálního hodnocení bude vypadat následovně:

$$d(x, h) = \left[\frac{1}{|I_x|}\sum_{i \in I_x}(x_i - h_i)^2\right]^{\frac{1}{2}}.$$

Tento vzorec opět předpokládá, že se věty nebudou odkazovat na hodnocení u jiných odstavců. Pro přidání této možnosti bude potřeba provést následující úpravy: Může nastat situace, kdy odstavec, na který se daná věta odkazuje, uživatel do svého textu nevybral. Pak jej také nemohl ohodnotit, čili není možné smysluplně vypočítat vzdálenost takovéto věty. Tato věta tedy

nemůže být doporučena a její vzdálenost bude nastavena na nejvyšší možnou. Pokud daný odstavec, na který se věta odkazuje, uživatel používá, pak se do vzorce přidá i výpočet vzdálenosti v této „nové dimenzi“ a musí se také odpovídajícím způsobem zvýšit konstanta, kterou se suma vydělí, aby zůstala dodržena definice průměrné vzdálenosti.

Další zvláštní situace vzniká u posledního odstavce. Zde se nachází pouze jediný atribut, který má reprezentovat výslednou známku. Na základě té se pak rozhoduje, zda je práce přijatelná či není. K tomu se váží odpovídající formulace, které jsou platné buď od hodnocení 50 (ekvivalent známky E) a výše, nebo naopak méně než 50. Pokud hodnocení spadá do správného intervalu, pak je věta zcela odpovídající a její vzdálenost by měla být nulová, v opačném případě věta neodpovídá a její vzdálenost by měla být maximální možná. Jako daný interval si tedy stanovím $[x_1, 100]$, pokud je atribut x_1 kladný, a $[0, x_1]$, pokud je atribut záporný. Vzdálenost pro poslední odstavec pak musím zavést tímto způsobem:

$$d_{overall}(x,h) = \begin{cases} 0, & (x_1 > 0 \wedge h_1 \geq x_1) \vee (x_1 < 0 \wedge h_1 < x_1), \\ \infty, & jinak. \end{cases}$$

Věty k poslednímu odstavci se však budou velmi často odkazovat na odstavce předchozí, bude pak nutné k této vzdálenosti přičíst vzdálenost od hodnocení v dalších požadovaných odstavcích, vypočtenou původním způsobem.

Při volbě metody výpočtu vzdálenosti celého shluku je potřeba si uvědomit, jaká situace by byla z pohledu uživatele „nejhorší“. Pokud mezi nabídnutými větami budou některé méně adekvátní, uživatel si je nemusí vybrat a dostane možnost nechat si místo nich doporučit jiné (viz následující sekce). Naopak pokud bude uživateli nabídnuto příliš málo vět k výběru, jeho možnosti se tím výrazně omezí, tato situace by tedy ideálně neměla nastat. Při výběru metody budu vycházet z metod diskutovaných v předchozí části textu. Jako příklad uvedu chování při dvou shlucích $A = \{a, b\}$ a $B = \{c, d\}$, kde $a = [50, -1]$, $b = [69, -1]$, $c = [-1, 60]$ a $d = [-1, 60]$, a aktuálním hodnocení $h = [70, 70]$.

- **Single-link**: Vzdálenost shluku C od aktuálního hodnocení definuji jako $D(C,h) = min\ d(x,h);\ x \in C$. V uvedeném případě by vzdálenosti vyšly takto: $D(A,h) = 1$, $D(B,h) = 10$.
- **Complete-link**: Vzdálenost $D(C,h) = max\ d(x,h);\ x \in C$. V tomto případě vychází vzdálenosti následovně: $D(A,h) = 20$, $D(B,h) = 10$.
- **Centroid**: Pokud bych používal centroid definovaný stejným způsobem jako v předchozí části, rovnala by se vzdálenost shluku vzdálenosti centroidu, tedy $D(C,h) = d(\mu(C),h)$. V ukázkovém případě bych tedy dostal $\mu(A) = [59.5, -1]$, $\mu(B) = [-1, 60]$ a $D(A,h) = 10.5$, $D(B,h) = 10$.

Věta *a* je aktuálnímu hodnocení jednoznačně nejbližší, proto by měla být doporučena na prvním místě. Podle navrženého chování se však první seřadí shluky podle celkové vzdálenost a až pak se ze shluku vybere nejvhodnější „kandidát“ k doporučení. Aby toto chování bylo co nejlépe odpovídající, je dobré zvolit jako vzdálenost shluku vzdálenost té věty, která z něj bude doporučena jako první. Skutečnost, že posléze mohou být doporučeny i méně odpovídající věty, nehraje velkou roli. V tomto konkrétním případě by při použití *complete-link* nebo *centroid* metody byla nejlépe odpovídající věta až na druhém místě. Použiji tedy metodu *single-link*, která požadovanému chování odpovídá nejlépe.

2.5 Doporučování a interaktivita

Když jsou všechny věty rozděleny do shluků podle jejich významu a shluky jsou seřazeny podle vypočtené vzdálenosti, může se přejít na samotné doporučení. Nyní popíši algoritmus doporučení a požadované reakce na vstup uživatele.

2.5.1 Doporučování vět

Abych mohl algoritmus popsat, definuji nejprve dvě konstanty, které budu následně využívat:

- **K**: Tato konstanta udává, kolik vět maximálně může být uživateli doporučeno v jednu chvíli. Jedná se o k v algoritmu kNN.
- **D**: Udává, jaká je maximální vzdálenost shluku, ze kterého budou vybírány věty k doporučení. Jinými slovy se jedná o hraniční vzdálenost, od které dále jsou již věty příliš vzdálené na to, aby je uživatel mohl využít.

V nejjednodušším případě by pak doporučení probíhalo následujícím způsobem:

1. Označím l jako pořadí prvního shluku se vzdáleností $> D$. Dále pak určím $k = min(K, l)$.
2. Z fronty shluků vyberu k nejbližších.
3. - Pokud je text odstavce zatím prázdný, vyberu z každého z k nejbližších shluků nejbližší větu s hodnotou $Position = 1$.
   - Pokud text odstavce prázdný není, vyberu z každého z k nejbližších shluků libovolnou nejbližší větu.
4. Vybrané věty doporučím uživateli k doplnění.

Uživatel však může v průběhu psaní posudku doporučené věty vybírat, měnit nebo odstraňovat. Po každé akci, která z doporučených vět některou vyjme, bude potřeba tento seznam doplnit. Dále tedy musím zavést dva příznaky u věty:

- **Recommended**: Označuje větu, která je uživateli právě doporučená, ale dosud ji nevybral.
- **Chosen**: Označuje větu, kterou si uživatel již vybral a začlenil do textu. Tento příznak budu také používat pro věty, které uživatel bez použití odstranil. Přestože jsem zamýšlel použít pro odstraněné věty jiný příznak, chování algoritmu doporučení by bylo v obou případech stejné, proto návrh takto zjednodušuji.

Shluk vět si pak označím jako **Active**, pokud obsahuje větu označenou jako *Recommended*. Z takto označeného shluku již nebudu doporučovat další věty. Tyto příznaky budou uloženy pouze ve vnitřní reprezentaci věty a shluku, do databáze se nijak nepromítnou.

Po některých událostech bude nutné zvýšit vzdálenost příslušného shluku o konstantu, kterou označím jako **ADD**. Vzdálenost shluku pak již nebude reprezentovat přesnou vzdálenost od aktuálního hodnocení, a proto by se měla společně se vzdáleností shluků měnit i konstanta D, aby nové změny ovlivňovaly pouze pořadí shluků a nezamezovaly použití některých vět po více vstupech uživatele.

S tímto ohledem bude potřeba podle původní hodnoty D před prvním doporučením rozdělit frontu seřazených shluků na dvě části a dále pak pracovat pouze s „vyhovující“ částí, dokud uživatel nezmění hodnocení. Využití konstanty D při doporučování pak ztrácí smysl. S ohledem na tyto změny pak upravuji algoritmus pro doporučování takto:

1. Označím l jako délku „vyhovující“ části fronty a r jako počet aktuálně doporučených vět. Dále pak určím $k = min(K - r, l)$.
2. Pro každý z k nejbližších shluků provedu následující:
   a) Pokud je shluk označený příznakem *Active*, žádnou větu ze shluku nevybírám a aktualizuji $k = min(k + 1, l)$.
   b) Pokud je text odstavce zatím prázdný, vybírám nejbližší větu s hodnotou $Position = 1$ a bez příznaku *Chosen*. Zde mám jistotu, že jsou použitelné pouze věty s hodnotou $Position = 1$, protože ostatní věty předpokládají jistý úvod do tématu, který se v prázdném odstavci nemůže nacházet. Pokud shluk takovou větu neobsahuje, nevybírám žádnou a aktualizuji $k = min(k + 1, l)$.
   c) Pokud text odstavce prázdný není, vybírám nejbližší větu bez příznaku *Chosen*. Pokud shluk takovou větu neobsahuje, nevybírám

žádnou a aktualizuji $k = min(k + 1, l)$. Hodnota *Position* zde již nebude hrát roli, protože automaticky zjistit, zda byly přesné předpoklady věty s hodnotou $Position = 2$ v dosavadním textu již splněny, by byl velmi náročný úkol. Dále omezovat podmínky, kdy větu s hodnotou $Position = 2$ doporučit, by mohlo uživatele nakonec jen omezovat v možnostech výběru, proto zde mohu tuto hodnotu již ignorovat.

3. Vybrané věty doporučím uživateli k doplnění a označím jako *Recommended*.

2.5.2 Reakce na vstup uživatele

Nyní popíši, jak by měla aplikace reagovat na akce uživatele v souvislosti s doporučováním vět:

- **Výběr věty do textu**: Po vybrání je věta odstraněna ze seznamu a příznak *Recommended* se mění na *Chosen*. Vzdálenost shluku se zvýší o konstantu ADD a shluky ve frontě se musí znovu seřadit. Pak se zavolá algoritmus doporučení, aby se zaplnilo „volné místo“ po větě.
- **Výběr alternativních znění**: Uživateli by mělo být umožněno zaměnit doporučenou větu za kteroukoliv jinou ze stejného shluku, případně do seznamu přidat z tohoto shluku více vět. Uživateli je tedy nabídnut seznam všech vět ve shluku, které nejsou označeny příznakem *Chosen*. Pokud se uživatel rozhodne danou větu zaměnit, původní věta je označena příznakem *Chosen* a všechny nově vybrané pak příznakem *Recommended*. Může pak nastat situace, kdy bude najednou doporučeno více vět než K. V takovém případě se pak po výběru sice spustí algoritmus doporučení, žádnou další větu však do seznamu nepřidá.
- **Odstranění věty ze seznamu**: Když se uživatel rozhodne odstranit danou větu bez výběru alternativního znění, je věta odstraněna ze seznamu a označena příznakem *Chosen*. Vzdálenost shluku se zvýší o konstantu ADD a shluky ve frontě se musí znovu seřadit. Pak se zavolá algoritmus doporučení, aby se zaplnilo „volné místo“ po větě.
- **Odstranění všech vět ze seznamu**: Dále bude mít uživatel možnost vymazat celý seznam doporučených vět a nechat si vygenerovat nové. Chování bude stejné jako v předchozím případě, jen se pro každou větu první nastaví příznak a odstraní ze seznamu, pak se zvýší vzdálenosti shluků a nakonec se jednou zavolá algoritmus doporučení.
- **Změna hodnocení**: Změní-li uživatel své hodnocení, je nutné znovu vypočítat vzdálenosti všech shluků vět, shluky znovu seřadit a novou frontu opět rozdělit podle konstanty D. Příznaky vět kromě *Recommended* zůstávají zachovány.

Kapitola 3

Implementace

V této kapitole diskutuji možné způsoby implementace a volbu implementace v podobě webové aplikace. Dále se věnuji vnitřní reprezentaci posudku v aplikaci, uživatelskému rozhraní a způsobu práce s aplikací. Nakonec popíši implementaci algoritmů navržených v předchozí kapitole.

3.1 Možné způsoby implementace

Navržené algoritmy by se jistě daly implementovat v desktopové aplikaci, která si lokálně drží svou databázi vět, které má nabízet. Při dnešních možnostech připojení k internetu však toho řešení není vhodné, protože nemá nadále cenu udržovat větší množství dat v lokální databázi a je také mnohem náročnější na údržbu. Také již samotná nutnost instalovat nový software začíná být nežádoucí, nejedná-li se o rozsáhlou aplikaci např. pro práci s multimediálními soubory.

Dalším možným řešením by bylo vytvořit rozšíření do již existujícího editoru, jako například Kile[10] nebo TEXmaker[11]. Tento způsob řešení má ovšem také své nevýhody. Jak jsem zmínil v předchozím odstavci, je nežádoucí držet všechna data lokálně, řešení tedy vždy bude vyžadovat připojení k internetu. Vzhledem k tomu, že k ostatním funkcím těchto editorů není připojení bezpodmínečně nutné (vyjmeme-li například instalaci nových balíčků), nebylo by vhodné editoru tuto výhodu odebrat.

Druhou nevýhodou je skutečnost, že vytvořit byť poměrně jednoduché rozšíření rozsáhlejší aplikace je časově mnohem náročnější než napsat aplikaci novou, protože je nutné se nejdříve podrobně seznámit s fungováním aplikace, kterou bude vytvářený nástroj rozšiřovat. Dále zde hraje roli také fakt, že by nebylo možné změnit toto řešení tak, aby bylo schopné pracovat s více formáty než pouze s formátem LATEX, a bylo by tedy méně flexibilní.

Jako nejvhodnější řešení tedy jednoznačně vyplývá vytvořit samostatnou webovou aplikaci, která v navržených krocích dovede uživatele k sestavení posudku. Tyto kroky budou základním rámcem, do kterého mohu navrho-

vané algoritmy zasadit. Musím však upozornit, že tato aplikace slouží zejména k ověření funkčnosti mého návrhu. Přestože jsem se snažil vytvořit aplikaci co nejlépe použitelnou, stále zdaleka není připravena k reálnému nasazení — toto by již překračovalo můj rozsah znalostí a není to požadováno ani v rámci mého oboru.

3.2 Použité technologie

Jako základní technologii jsem použil programovací jazyk Java, konkrétně jeho nejnovější podobu Java 8. Zvolil jsem tento jazyk, protože je velmi flexibilní, existuje k němu velké množství různých rozšíření, která podporují tvorbu webových aplikací, a zároveň mám s tímto jazykem již určité zkušenosti, díky kterým byl vývoj aplikace znatelně pohodlnější a rychlejší.

Jako webovou technologii pro jazyk Java jsem použil framework Spark[12], který je přímo vytvořený pro verzi Java 8, konkrétně využívá lambda výrazy, které v předchozích verzích podporovány nebyly. Díky tomu zaručí stejnou funkčnost daleko menší množství kódu, než by tomu bylo u jiných technologií. Spark je přímo navržený pro rychlý a jednoduchý vývoj webových aplikací, což z něj dělá v mém případě ideální volbu.

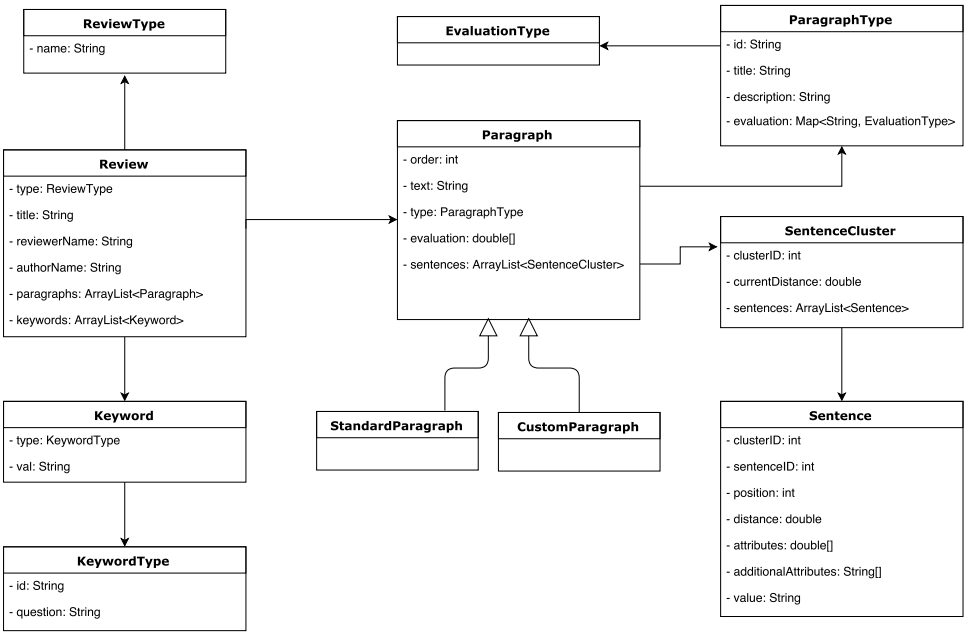
Pro tvorbu webových stránek a skriptů z Java aplikace jsem použil Velocity Template Engine[13], který je také výchozím nástrojem pro Spark. Abych mohl ukázat, že návrh podporuje dostatečnou interaktivitu, používám také technologie JavaScrpit, jQuery a jQuery UI, které umožní dynamicky reagovat na vstup uživatele přímo ve webovém prohlížeči. Pro tvorbu vizuální stránky uživatelského rozhraní jsem použil styl Bootstrap a dvě rozšíření: Bootstrap Slider[14] a Tag-it[15]. U stažených souborů se vždy nachází v příslušném adresáři také textový soubor s licencí.

3.3 Reprezentace posudku v aplikaci a průběh jeho tvorby

Zde se budu věnovat základní funkčnosti aplikace. Vysvětlím reprezentaci posudku pomocí tříd a popíši jednotlivé kroky sestavení posudku pomocí této aplikace.

3.3.1 Vnitřní reprezentace posudku

Aplikaci jsem se snažil navrhnout tak, aby po případném odebrání uživatelského rozhraní a tříd, které se k němu váží, zůstalo univerzálně použitelné API, které by umožňovalo vytvořit libovolným způsobem aplikaci s navrhovanou funkčností. Jedná se konkrétně o balíčky **review** a **review.sentences**, které jsou v mé implementaci stěžejní. Nyní popíši třídy, které tyto balíčky obsahují, pro názornost je zobrazuji také v diagramu 3.1.



Obrázek 3.1: Diagram tříd reprezentujících posudek

Třída **Review** reprezentuje celý posudek. Obsahuje seznam použitých odstavců a seznam klíčových slov, která popisují téma, kterého se posuzovaná práce týká. Dále je určen typ, možnosti jsou obsaženy ve výčtu ReviewType, kde prozatím nabízím možnosti „bachelor thesis“ a „master thesis“, tedy bakalářská a diplomová práce. Tyto typy neovlivňují nic v dalších fázích tvorby posudku, protože v posudcích bakalářských a diplomových prací se objevují většinou podobné fráze. Pokud by však bylo žádoucí rozšířit funkčnost i na jiné druhy posudků, může se z typu práce stát další indicie pro identifikaci odpovídajících formulací. Použitím výčtu také mohu přehledně specifikovat, které možnosti jsou současně podporovány a tuto informaci jednoduše převést do uživatelského rozhraní.

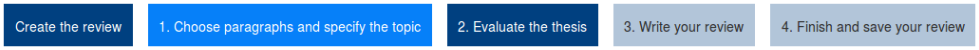
Třída **Paragraph** reprezentuje jeden odstavec textu. Jedná se o polymorfní typ, který zahrnuje dvě podtřídy — StandardParagraph a CustomParagraph. Z hlediska doporučování vět je zajímavá hlavně třída StandardParagraph, která reprezentuje jeden z odstavců tak, jak jsem je definoval ve struktuře posudku. Výčet těchto odstavců obsahuje ParagraphType, kde je ke každému z definovaných odstavců přiděleno ID, nadpis, krátký popis, čeho se odstavec týká, a seznam otázek, které se k němu váží. Výčet druhů hodnocení jsem opět pro přehlednost seskupil do typu EvaluationType. Každý odstavec si také drží seznam doporučených shluků vět. Třídu CustomParagraph jsem zde přidal, aby byla podporována možnost začlenit do textu i jiný než standardní

odstavec, ve své ukázkové aplikaci ji však nepoužívám.

Balíček **sentences** obsahuje třídy, které se váží k doporučování vět a ke klíčovým slovům. Klíčová slova reprezentuje třída **Keyword**, čtyři druhy klíčových slov společně s otázkami, které se k ním váží, jsou opět dány ve výčtu KeywordType. Jednotlivé věty jsou zastoupeny pomocí třídy **Sentence**, tak odpovídá položce věty v databázi, obsahuje tedy pozici v textu, ID shluku, do kterého patří, atributy pro výpočet vzdálenosti a navíc ještě ID této věty a aktuální vzdálenost. Tyto dvě hodnoty se do databáze neukládají, protože mají význam vždy pouze v rámci jednoho posudku. Celý shluk vět pak reprezentuje třída **SentenceCluster**, která obsahuje seznam vět ve shluku, aktuální vzdálenost shluku a ID shluku.

3.3.2 Uživatelské rozhraní a tvorba posudku v krocích

Aby byla práce s aplikací pro uživatele srozumitelná, rozdělil jsem proces tvorby posudku v předchozí kapitole do pěti částí. V aplikaci jsem pak vytvořil navigační lištu, pomocí které se lze mezi jednotlivými kroky volně pohybovat.

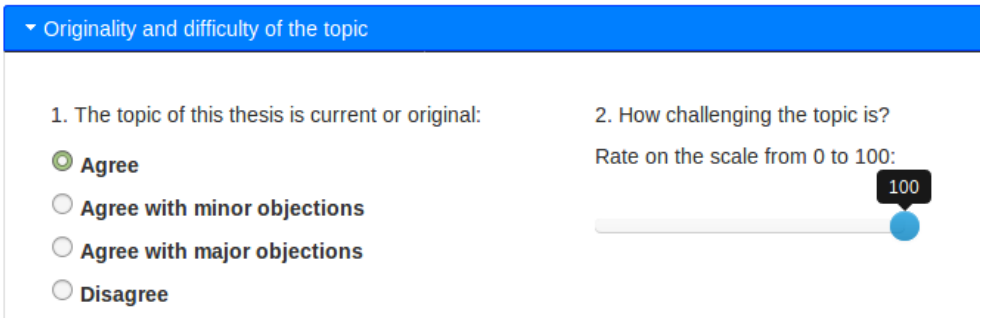


Obrázek 3.2: Navigační lišta pro přecházení mezi jednotlivými kroky

Na obrázku 3.2 jsou na navigační liště vidět tlačítka ve třech barvách. Světle modrou je označený krok, ve kterém se uživatel právě nachází. Tmavě modrou jsou označeny kroky, na které se uživatel může přesunout a šedé jsou kroky, pro které dosud nejsou splněny všechny předpoklady. Není možné například přejít rovnou k psaní posudku, aniž by uživatel vybral odstavce, které bude používat, a práci ohodnotil.

„Nultý“ krok, čili úvodní strana aplikace, by měla být velmi jednoduchá, aby se uživatel v prostředí mohl zorientovat. Obsahuje tedy pouze pole pro vyplnění názvu a druhu posuzované práce a jmen autora a hodnotitele. Nachází se zde také odkaz na stránku, ze které je možné přidávat do databáze nové věty. Tu jsem vytvořil pro sebe jako pomůcku při sestavování databáze a pro ukázku ji v implementaci ponechávám. V prvním kroku pak uživatel vybírá ze seznamu, který vychází z hodnot definovaných ve výčtu ParagraphType, odstavce pro svůj text, a zároveň má možnost pomocí čtyř otázek daných ve výčtu KeywordType zadat klíčová slova. Klíčová slova však nejsou pro další průběh nijak podstatná, tento krok tedy může uživatel bez problému přeskočit.

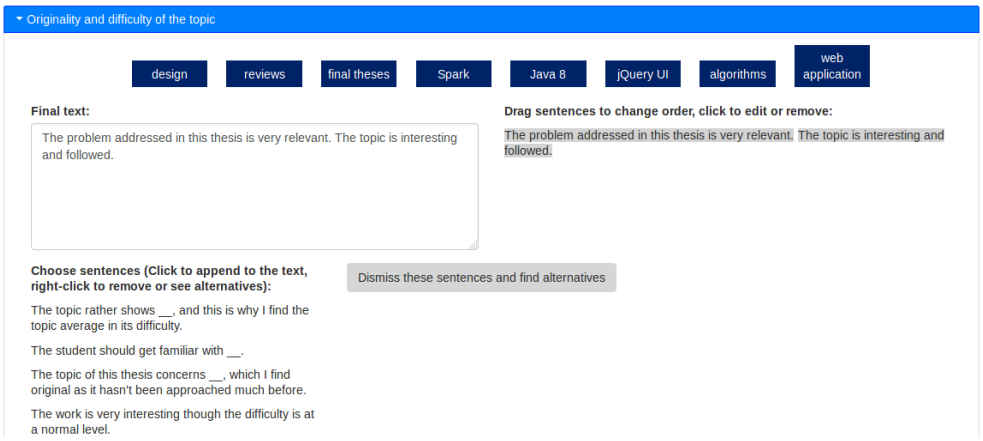
Ve druhém kroku je uživatel vyzván k ohodnocení práce. Otázky k jednotlivým hodnoceným aspektům jsou definovány opět ve výčtu ParagraphType a do uživatelského rozhraní se promítnou způsobem zobrazeným na obrázku 3.3.



Obrázek 3.3: Hodnocení

Pro pokračování do dalšího kroku tvorby posudku je nutné, aby byly zodpovězeny všechny otázky. Proto má každá otázka nastavenou výchozí odpověď, kterou pak uživatel mění podle svého uvážení. Nezmění-li uživatel odpověď, počítá se s optimistickým scénářem a výchozí odpovědi jsou vždy nejlepší možné (nejvíce pozitivní).

Po zadání kompletního hodnocení se může spustit výpočet vzdáleností a algoritmus pro doporučování, jejichž implementaci popíši posléze. Ve třetím kroku pak může uživatel sepsat svůj posudek za pomoci doporučených vět.



Obrázek 3.4: Uživatelské rozhraní ve třetím kroku

Na obrázku 3.4 je zobrazená část uživatelského rozhraní, ve které uživatel tvoří text jednoho odstavce. V horní části jsou klíčová slova, která se po kliknutí doplní do právě aktivního textového pole. V levé části se nachází textové pole pro kompletní text odstavce, pod ním pak doporučené věty, které se po kliknutí přidají na konec textu. Pokud by chtěl uživatel větu zařadit na jiné místo, může ji pak přemístit v pravé části, kde se dají věty tažením řadit, dále

také zvlášť upravovat nebo odstraňovat z textu.

Po kliknutí levým tlačítkem na doporučenou větu, která není kompletní, se zobrazí dialogové okno, ve kterém uživatel může doplnit chybějící části věty, případně využít klíčová slova. Po kliknutí pravým tlačítkem se zobrazí v dialogovém okně seznam všech vět ve shluku a uživatel může vybrat libovolný počet z nich, nebo nabízenou větu odstranit a žádnou alternativu k ní nevybrat. Tlačítko napravo od doporučených vět slouží k „zamíchání“ výběru, všechny doporučené věty odstraní a pokud existují další vyhovující, doplní z nich požadovaný počet na místo předchozích.

V posledním kroku po kontrole a případné úpravě textu má uživatel možnost uložit si svůj posudek v podobě zdrojového kódu pro LaTeX. Tento kód je stejně jako kód všech stránek této aplikace generován pomocí Velocity Template Engine, z pohledu algoritmů tedy tento krok již příliš zajímavý není.

3.4 Identifikace vhodných formulací

Nyní mohu přejít k implementaci samotných algoritmů pro identifikaci vhodných formulací. Upřesním způsob uložení vět v databázi a výpočtu vzdáleností definovaných v předchozí kapitole.

3.4.1 Databáze vět

Pro uložení vět v databázi používám soubory typu CSV. Jedná se o velmi jednoduchou formu, kde řádky textového souboru reprezentují položky v databázi a jednotlivé atributy jsou odděleny zvoleným znakem, nejčastěji středníkem (;). Tento formát je pro mé potřeby naprosto dostačující. Věty v databázi jsou rozděleny po odstavcích, ke kterým patří, každý odstavec má tedy svůj CSV soubor s databází vět, jehož jméno je identické s identifikátorem odstavce, definovaným ve výčtu ParagraphType. Uložená věta v databázi pak může vypadat takto:

```
Cluster;Position;Attr0;Value
11;2;50.0;04fulfilment.3.0~While having no publishable value, it deserves grade $g.
```

Schválně jsem zde vybral větu z databáze k poslednímu odstavci, která obsahuje odkaz na hodnocení jiného odstavce. Tyto odkazy se nachází až za posledním středníkem, takže jsou z databáze přečteny společně s „hodnotou“ věty. Mezi sebou a od samotné věty jsou pak odkazy odděleny znakem ~. Odkaz se skládá ze tří částí oddělených tečkou: na začátku je identifikátor odstavce, na který se atribut odkazuje (04fulfilment), za ním pořadí otázky (3) a nakonec hodnota tohoto atributu (0). Věta také obsahuje speciální znak „$g“, který je těsně před doporučením nahrazen odpovídající známkou podle aktuálního hodnocení na škále A–E.

Čtení z databáze zajišťuje třída **SentenceDBHandler**. Je užitečné vytvářet vždy pouze jednu instanci této třídy, která bude provádět veškeré čtení a zápis, proto zde používám návrhový vzor Singleton [16]. Záznam v databázi se převede na instanci třídy **Sentence**, která je deklarovaná tímto způsobem:

```
public class Sentence {
        private int clusterID;
        private int sentenceID;
        private int position;
        private double distance;
        private double[] attributes;
        private String[] additionalAttributes;
        private String value;
        ...
}
```

Jednotlivé členské proměnné odpovídají záznamu věty v databázi, navíc se zde nachází pole *additionalAttributes*, ve kterém se nachází případné odkazy na hodnocení jiných odstavců, které jsou ponechány v textové podobě stejně jako v databázi. Do proměnné *distance* se ukládá aktuální vzdálenost věty po každém výpočtu.

3.4.2 Shlukování podobných vět

Shluk vět je reprezentován třídou **SentenceCluster**, která je deklarovaná velmi jednoduše jako:

```
public class SentenceCluster {
        private int clusterID;
        private double currentDistance;
        private ArrayList<Sentence> sentences;
        ...
}
```

Po načtení vět z databáze je každá instance třídy Sentence, která obsahuje v proměnné *clusterID* hodnotu -1, zařazena do nové instance třídy SentenceCluster. Pokud k takové situaci dojde, spustí se algoritmus pro hierarchické shlukování, který nové věty rozdělené do jednočlenných shluků zařadí do existující struktury shluků (některé shluky samozřejmě mohou zůstat nové). Toto zajišťuje metoda **recountClusters** třídy **SentenceDBHandler**. Minimální požadovanou vzdálenost dvou shluků udává statická proměnná MIN_CLUSTER_DISTANCE. Figuruje zde také metoda **countDistance**, která implementuje výpočet vzdálenosti mezi dvěma shluky tak, jak jsem ji definoval v předchozí kapitole. Vzhledem k úpravě, která dovoluje atributy vět odkazující se na hodnocení jiných odstavců, jsou kódy poměrně dlouhé a

nebudu je zde uvádět, v případě zájmu je kompletní kód dostupný v příloze této práce. Připomínám pouze, že před výpočtem vzdálenosti mezi dvojicí vět je nejprve potřeba porovnat, na které atributy se odkazují, a pouze pokud se tyto atributy shodují, může být vzdálenost vypočtena, v opačném případě je vrácena vzdálenost maximální.

3.4.3 Výpočet vzdálenosti od aktuálního hodnocení

Výpočty vzdáleností shluků vět od aktuálního hodnocení zajišťuje třída **SentenceRecommender**. Stejně jako při čtení databáze, ani zde není nutné vytvářet více instancí této třídy, proto je také implementována podle návrhového vzoru Singleton. Tato třída obsahuje kromě metody getInstance (vycházející ze Singleton návrhu) jedinou veřejnou metodu **recommend**.

Tato metoda je definována pro dva případy: pokud je jako argument poskytnutý seznam shluků, které se mají seřadit podle vzdálenosti, nenačítají se shluky z databáze. Pokud tento seznam poskytnutý není, pak se shluky načtou pomocí třídy SentenceDBHandler a následně se seřadí podle vzdáleností. Tato metoda vždy vrací seznam instancí třídy SentenceCluster seřazený podle vzdálenosti k aktuálnímu hodnocení.

Vzhledem ke všem úpravám, které jsem v předchozí kapitole definoval, je kód této metody opět poměrně dlouhý. Nebudu jej zde uvádět a základní funkčnost popíši pouze slovně.

Na počátku si deklaruji výsledný seznam shluků jako prázdný seznam. Poté po případném načtení shluků z databáze je nutné pro každý shluk zkontrolovat, zda jsou splněny podmínky pro výpočet vzdálenosti. Shluk musí obsahovat alespoň jednu větu a zároveň pokud se odkazuje na hodnocení jiných odstavců, tyto odstavce musí být v posudku ohodnoceny. Pokud shluk neobsahuje žádnou větu, může se rovnou přejít k dalšímu shluku v seznamu. Pokud se shluk pouze neshoduje v odkazech na další hodnocení, je nastavena vzdálenost na maximální možnou, ale k příslušnému odstavci je uložen pro případ, že by uživatel později daný odstavec ohodnotil a věty ze shluku by pak mohly být doporučeny.

Po kontrole podmínek se pak podle typu odstavce rozhodne, který ze dvou vzorců pro výpočet vzdálenosti je možné využít. Při výpočtu se ukládá vzdálenost každé věty a pak i shluku do příslušných instancí tříd. Po výpočtu vzdálenosti se pak daný shluk uloží k příslušnému odstavci a pokud je vzdálenost shluku vyhovující (hranice je stanovena statickou proměnnou MAX_DISTANCE), pak je uložen do výsledného seznamu shluků k doporučení. V každém shluku se také seřadí věty podle vzdáleností. Po výpočtu vzdáleností všech shluků se seřadí i shluky ve výsledném seznamu, přičemž shluky a věty ve shlucích se stejnou vzdáleností jsou seřazeny náhodně, aby nebyly doporučovány stále stejně.

3.5 Doporučování a interaktivita

Tento seřazený seznam obsahuje pouze věty, jejichž vzdálenost je určena jako vyhovující. Navrhovanou konstantu D tedy reprezentuje statická proměnná $MAX\_DISTANCE$ třídy *SentenceRecommender*. Tyto vyhovující seřazené shluky jsou pak odeslány na stranu klienta na počátku třetího kroku tvorby posudku. Samotné interaktivní doporučování pak probíhá pouze na straně klienta. Ve skriptu **step3-recommender.js** definuji funkci **recommend**, která implementuje doporučení popsané v předchozí kapitole.

V tomtéž skriptu se nachází kód pro všechny definované reakce na vstup uživatele. Dále pak skript **step3-sentences.js** zajišťuje doplňování, přesouvání a úpravu vět v textu posudku, skript **step3-clusters.js** definuje strukturu objektů reprezentujících věty a shluky. Tyto objekty vychází z podoby tříd v serverové části aplikace a jsou rozšířeny o příznak **cActive** v případě shluku a příznaky **sChosen** a **sRecommended** v případě věty. Skript **step3-keywords.js** zajišťuje doplňování klíčových slov na příslušné místo v posledním aktivním textovém poli.

Kapitola 4

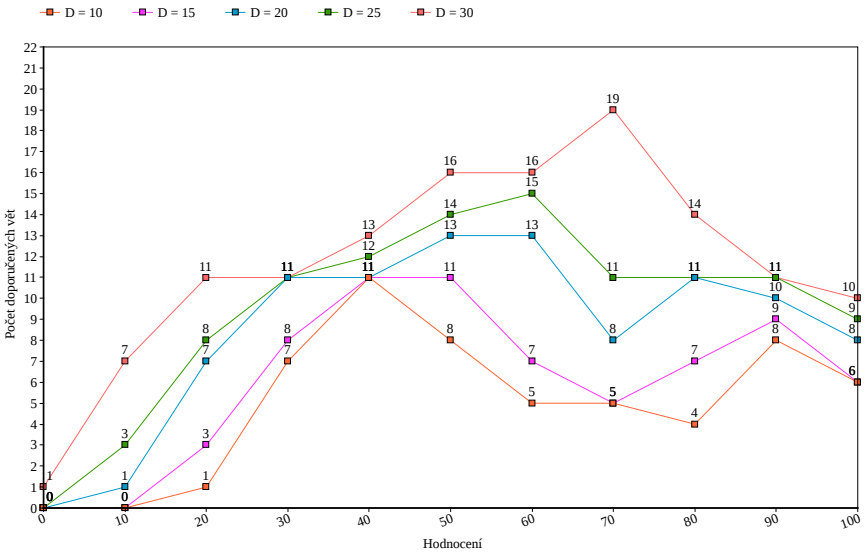
Výsledky

V této kapitole hodnotím výsledky, kterých navrhované algoritmy dosahují nad konkrétní databází. Ukazuji zde také jako výsledek práce s aplikací dva posudky, které jsem pomocí aplikace pro ověření sestavil. Jako předlohu jsem použil posudky dostupné v informačním systému FIT ČVUT, které jsem zároveň nevyužíval jako předlohu k sestavení databáze.

4.1 Databáze

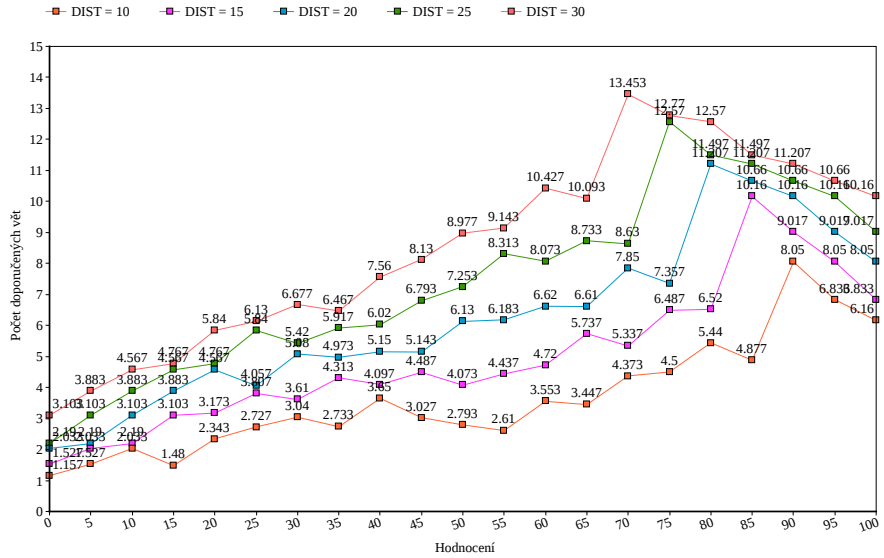
Prvním předpokladem pro dosažení požadovaných výsledků je databáze, která bude pokrývat všechna dostupná hodnocení a pro většinu situací nabídne dostatečné množství formulací, jak tato hodnocení vyjádřit. Abych tyto kvality v databázi mohl vyjádřit, provedl jsem pro soubor vět ke každému odstavci tento výpočet: Pro každý atribut daného odstavce a pro hodnoty maximální vzdálenosti $D = 5, 10, 15, 20, 25$ a 30 jsem zjišťoval, kolik vět, které mají hodnotu tohoto atributu jinou než -1, by mohlo být při různých hodnoceních na základě tohoto atributu doporučeno. Pro příklad uvádím v grafu 4.1 počty doporučených vět na základě 1. atributu ve 3. odstavci.

Věty u ostatních atributů jsou rozděleny velmi podobně. Je patrné, že pro negativní hodnocení databáze obsahuje výrazně nižší počet vět než pro hodnocení kladná, což může být způsobeno dvěma faktory. Negativní posudek by měl obsahovat konstruktivní kritiku, využití univerzálních formulací je zde tedy velmi omezené a hodnotitel bude pravděpodobně komentovat spíše konkrétní důvody, proč práci hodnotí negativně. Zjevně jsem také do sady posudků, ze kterých jsem vybíral věty, zařadil příliš málo posudků negativních, a tedy má představa, že určitý vzorek posudků vybraný z informačního systému fakulty bude dostačující, se ukázala jako mylná. První z těchto faktorů nijak ovlivnit nemohu a vět týkajících se negativního hodnocení bude vždy méně, avšak neměla by nastávat situace, že pro určité hodnocení nebude k dispozici žádná věta. Proto jsem musel rozšířit databázi o vlastní věty tak, aby obsahovala alespoň jednu možnost vyjádření každého hodnocení. Takto doplněná data-



Obrázek 4.1: Počty doporučených vět: atribut 1, odstavec 3

báze obsahuje 281 vět, průměrně na odstavec vychází 46.8 vět a pro ukázku funkčnosti algoritmů by měla být dostačující. V grafu 4.2 zobrazuji průměrný počet doporučených vět na základě jednoho atributu pro různá hodnocení a různé hodnoty maximální vzdálenosti doporučené věty (D). Pro vzdálenosti $D \geq 15$ je splněna podmínka alespoň jedné věty pro každé hodnocení.

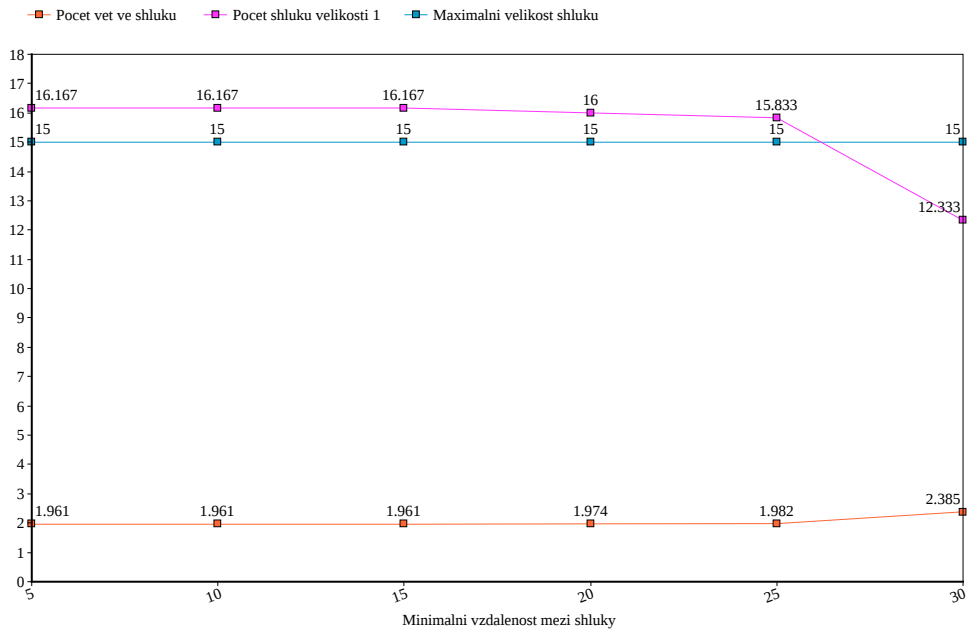


Obrázek 4.2: Průměrný počet doporučených vět v závislosti na hodnocení

4.2 Shlukování

Dále jsem zjišťoval, jakých výsledků dosahuje algoritmus shlukování. Důležité jsou zde dva aspekty. Shluk by měl obsahovat dostatek alternativ pro vyjádření každého hodnocení, ale nikdy by neměl obsahovat dvě věty s rozdílným významem, protože předpokládám, že věty ve shluku budou vzájemně zaměnitelné. Graf 4.3 zobrazuje závislost tří hodnot na volbě minimální vzdálenosti mezi dvěma shluky.

- **Počet vět ve sluku** vyjadřuje průměrný počet vět v jednom shluku.
- **Počet shluků velikosti 1** vyjadřuje průměrný počet shluků k jednomu odstavci obsahujících jedinou větu.
- **Maximální velikost shluku** vyjadřuje celkové maximum v počtu vět v jednom shluku.



Obrázek 4.3: Výsledky algoritmu shlukování

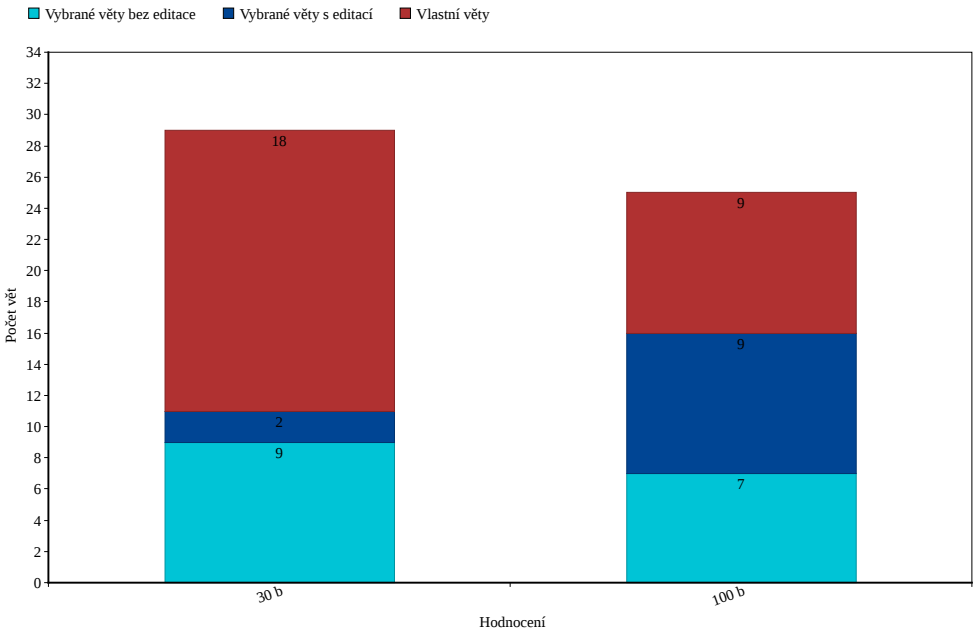
Z grafu vyplývá, že požadavek na velikost shluků splněn není, což je způsobeno zejména velikostí databáze. Je zřejmé, že hodnota minimální vzdálenosti mezi shluky výsledky téměř neovlivňuje, zásadní roli zde tedy budou hrát podmínky, kdy věty ve společném shluku být nemohou a jejich vzájemná vzdálenost je nastavena na maximální. Tyto podmínky jsou velmi restriktivní, ale

jsou také nutné. Bez nich by se ve shlucích velmi často objevovaly věty s odlišným významem. Algoritmus jistě dovoluje splnit i požadavek na velikost shluků, byla by však k tomu potřeba mnohem robustnější databáze, než jakou jsem dokázal pro ukázku funkčnosti sestavit.

4.3 Analýza posudků sestavených pomocí aplikace

Pro ověření funkčnosti celé ukázkové aplikace jsem pomocí ní sestavil dva posudky, které jsou kompletní k nalezení v příloze A této práce. V posudcích jsem označil tři druhy vět. Věty podtržené plnou čarou jsou vybrány z doporučených vět bez změny. Věty podtržené tečkovaně jsou také vybrány z doporučených vět, ale před doplněním do textu vyžadovaly určitou míru editace. Věty, které nejsou podtržené, jsem do textu doplnil sám.

V informačním systému FIT jsem vybral dva rozdílné posudky, které jsem nevyužíval k sestavení databáze, a pomocí aplikace jsem se snažil stejné hodnocení vystihnout v angličtině. Prvním z nich je posudek Ing. Radka Hronzy k poprvé odevzdané práci *Analýza možnosti elektronizace podnikových procesů*[17], která nebyla obhájena a vedoucí v posudku navrhuje hodnocení 30 body. Druhým je posudek Ing. Josefa Kokeše na práci *Analýza kryptoviru*[18], který navrhuje nejvyšší možné hodnocení 100 bodů. Posudky jsem vybral tak, aby se svým hodnocením výrazně lišily, jinak byl výběr čistě náhodný.



Obrázek 4.4: Poměr vybraných a vlastních vět v sestaveném posudku

Jako maximální vzdálenost shluku vět, který ještě může být doporučen, jsem zvolil hodnotu 25. Při této hodnotě mám již zaručeno doporučení poměrně široké škály formulací, ale stále se mezi nimi neobjevilo mnoho takových, které by zadanému hodnocení vůbec neodpovídaly. Pro první (negativní) posudek bylo doporučeno celkem 142 vět v 54 shlucích, pro druhý (kladný) posudek 188 vět v 69 shlucích.

V grafu 4.4 zobrazuji poměry doporučených vět, které jsem začlenil do posudku, a vět vlastních. Je patrné, že v negativním posudku je vybraných vět v poměru k vlastním výrazně méně než v posudku kladném. Tento výsledek odpovídá předpokladům, které zmiňuji při analýze počtu doporučených vět v závislosti na hodnocení. Při sestavování těchto posudků jsem se samozřejmě snažil maximalizovat využití doporučovaných vět, ale jak z grafu vyplývá, v ideálním případě se dá smysluplný posudek sestavit téměř ze dvou třetin na základě nabízených formulací, a to s poměrně omezenou databází vět. Při rozšíření této databáze lze pak předpokládat lepší výsledky.

Názor, zda tato aplikace urychlí psaní posudku, je samozřejmě subjektivní. Při práci s ní jsem si však, že díky doporučeným větám jsem strávil méně času rozmýšlením, jak přesně danou myšlenku vyjádřit. První z posudků jsem navíc psal při cestě vlakem, tedy v prostředí, kde jsem byl často rozptylován různými podněty. Navrhované věty mi pak pomáhaly „neztratit myšlenku“ a v případě rozptýlení si snáze vzpomenout, jak jsem chtěl v psaní textu pokračovat.

Závěr

V této práci se věnuji návrhu algoritmů, které podporují tvorbu posudku kvalifikační práce zejména pomocí doporučování formulací, které může uživatel ve svém posudku využít.

V první kapitole popisuji strukturu posudků, které sloužily jako vzor, a navrhuji vlastní strukturu posudku tak, aby tvořil souvislejší text. Dále v této kapitole zmiňuji několik aplikací, které se svou funkcionalitou blíží mému návrhu, avšak žádné aplikace poskytující podporu při psaní posudku se mi nalézt nepodařilo. Ve druhé kapitole popisuji proces tvorby posudku rozdělený do pěti kroků a navrhuji algoritmy, které v předposledním kroku sestavení posudku mohou na základě hodnocení uživatele identifikovat a doporučit vhodné formulace k začlenění do textu. Také se zde věnuji způsobu identifikace podobných formulací, které mohou být navzájem zaměnitelné. Dále pak navrhuji způsob, jakým by měly být realizovány reakce na vstup uživatele.

Ve třetí kapitole diskutuji možnosti, jak by navrhované algoritmy mohly být implementovány, zdůvodňuji volbu implementace pomocí webové aplikace. Popisuji pak implementaci pomocí konkrétních technologií, způsob reprezentace posudku v datových strukturách a konkrétní podobu doporučování vět v uživatelském rozhraní. V poslední kapitole pak analyzuji sestavenou databázi vět, počty doporučených vět při různém hodnocení a počty vět identifikovaných jako podobné. Nakonec uvádím dva příklady posudků sestavených pomocí aplikace, svůj subjektivní pocit z používání aplikace a diskutuji možné výhody, které mé řešení při sestavování posudku může poskytnout.

Ukázalo se, že v mém případě se čas potřebný k sestavení posudku za použití aplikace zkrátil. Aplikace mi napomohla rychleji zformulovat myšlenky a neopomenout žádný hodnocený aspekt. Věřím tedy, že ačkoli je sestavení posudku subjektivní záležitost a každý uživatel by si podporu mohl představovat jinak, navržené algoritmy jsou využitelné a alespoň některým by práci mohly usnadnit.

Literatura

[1] Cormode, G.: How NOT to review a paper, The tools and techniques of the adversarial reviewer. *SIGMOD Record*, ročník 37, č. 4, 12. 2008. Dostupné z: `https://sigmodrecord.org/publications/sigmodRecord/0812/p100.open.cormode.pdf`

[2] Basu, A.: A tutorial on how to use Gradepro GDT Tool for writing your reviews. *PeerJ Preprints*, 10. 2016. Dostupné z: `https://peerj.com/preprints/2520.pdf`

[3] McMaster University, 2015 (developed by Evidence Prime, Inc.): *GRADEpro GDT: GRADEpro Guideline Development Tool [Software]*. 2017, [cit. 2017-04-30]. Dostupné z: `https://gradepro.org/`

[4] Friesike, S.; Bartling, S.; Fecher, B.; aj.: Openingscience.org — Software & Tools. [cit. 2017-04-30]. Dostupné z: `http://www.openingscience.org/category/initiatives/software-tools/`

[5] Ebrahim, N. A.: Research Tools: Scientific Writing Tools for Writing Literature Review and a Paper. 2016, [cit. 2017-04-30]. Dostupné z: `https://www.mindmeister.com/39583892/research-tools-by-nader-ale-ebrahim`

[6] March, T.: The Original Persuasive Essay Maker. 1995, [cit. 2017-04-30]. Dostupné z: `http://www.ozline.com/electraguide/thesis.php`

[7] Peterson, L. E.: K-nearest neighbor. *Scholarpedia*, ročník 4, č. 2, 2009: str. 1883, revision #136646, [cit. 2017-05-06]. Dostupné z: `http://www.scholarpedia.org/article/K-nearest_neighbor`

[8] Gan, G.; Ma, C.; Wu, J.: *Data Clustering: Theory, Algorithms and Applications*. ASA-SIAM Series on Statistics and Applied Probability, SIAM, Philadelphia, ASA, Alexandria, VA, 2007, ISBN 978-0-898716-23-8.

[9] The Pennsylvania State University, Eberly College of Science: *Agglomerative Hierarchical Clustering*. 2017, [cit. 2017-05-02]. Dostupné z: https://onlinecourses.science.psu.edu/stat505/node/143

[10] Wijnhout, J.; Ludwig, M.: *Kile - an Integrated LaTeX Environment*. [cit. 2017-05-07]. Dostupné z: http://kile.sourceforge.net/

[11] Brachet, P.: *TeXmaker, The universal LaTeX editor*. [cit. 2017-05-07]. Dostupné z: http://www.xm1math.net/texmaker/

[12] Wendel, P.; Åse, D.: *Spark Framework: A tiny Java web framework*. [cit. 2017-05-08]. Dostupné z: http://sparkjava.com/

[13] The Apache Software Foundation: *The Apache Velocity Project*. 2016, [cit. 2017-05-08]. Dostupné z: http://velocity.apache.org/

[14] Kemp, K. J.: *Slider for Bootstrap*. [cit. 2017-05-08]. Dostupné z: https://github.com/seiyria/bootstrap-slider

[15] Ehlke, A.: *Tag-it: a jQuery UI plugin*. [cit. 2017-05-08]. Dostupné z: https://github.com/aehlke/tag-it

[16] Geary, D.: Simply Singleton. *Java World*, 2003, [cit. 2017-05-09]. Dostupné z: http://www.javaworld.com/article/2073352/core-java/simply-singleton.html

[17] Kim, A.: *Analýza možnosti elektronizace podnikových procesů*. Bakalářská práce, České vysoké učení technické v Praze, Fakulta informačních technologií, 2016.

[18] Řečínský, J.: *Analýza kryptoviru*. Diplomová práce, České vysoké učení technické v Praze, Fakulta informačních technologií, 2015.

Příloha **A**

Posudky sestavené pomocí aplikace

Analysis of possibilities of electronization of business processes
Bachelor thesis review
Alexey Kim

Jakub Vašíček (based on the review of Ing. Radek Hronza)

14/05/2017

1 Originality and difficulty of the topic

I find the topic average in its difficulty.

2 Factual level of the thesis and bibliography

The written thesis is sufficient in size, but the text is too brief and some parts should have been more detailed. The text lacks detailed description that can be considered a demonstration of the electronization process. I also miss any note on the benefits of modifications of the process portal for the organization's management. The work with sources of information is also very weak. I have found only one citation from the page 7 on. Overall, the thesis lacks important pieces of information and context.

3 Formal level of the written thesis

I could find a large number of significant grammatical errors in the text. I understand that the author is not a native Czech speaker. However, he ignored all my suggestions to get a correction from someone more competent. As for the logical structure — the thesis could be better organized and structured. For a reader who is not familiar with the topic the text is very hard to read.

1

4 Fulfilment of the assignment and applicability of the results

The goal was reached only in certain aspects and not as a whole. I am not fully satisfied with fulfilment of the point 3. Points 4, 5 and 6 are also described too briefly. It is a work which does not have a deeper critical analysis and at this stage the results have no real use.

5 Activity and self-reliance of the student

The student regularly attended the consultation meetings. However, I am disappointed to see that he ignored my recommendations and many outcomes of our consultations were not incorporated in the work.

6 Overall evaluation

Considering what has been said above, I evaluate this thesis with grade F and I do not recommend it for defence. I do not claim that the author did not do sufficient amount of work. However, the text of this thesis does not seem so. The description of the process of electronization is very limited. The design of modifications is also too brief and not useful as it is. The description of results and recommendation is insufficient. Furthermore, me and my colleague could not find a proof of electronization of the given process on the on the attached CD. All we found there was only unfinished process diagram. Therefore, I suspect the author of not realizing the electronization at all or at least not finishing it. Or he did not attach tthe documents that would allow to run the electronized process. From my point of view I must advise the student to rework this thesis.

Analysis of a cryptovirus
Master thesis review
Bc. Jan Řečínský

Jakub Vašíček (based on the review of Ing. Josef Kokeš)

11/05/2017

1 Originality and difficulty of the topic

The task undertaken by the student in this thesis is a challenging one. Reverse analysis of software is a complex topic and with malware it gets even more complicated, as malware is made to be resistant to analysis. The student had to put a great effort to reach any point of the analysis, which is more than a student can be asked for in a final thesis.

2 Factual level of the thesis and bibliography

I have no objections to the factual level of this thesis. The size of the work corresponds to the expected level and the work with sources of information was significantly improved. The student based his work on relevant sources, however the choice may sometimes seem a little strange. For example, in the part regarding TOR, I would expect an official website instead of a third person's blog. These articles and used bibliography are properly cited. As a reader I would welcome references to articles concerning reconstruction of the program from its image in memory, but this is not a crucial topic to this work.

3 Formal level of the written thesis

The content of the thesis is logically divided into chapters and I found only two minor problems. The last paragraph of the chapter 1.1 does not make sense in this place (this is the consequence of late refactoring before

the deadline). Chapters 5.2.8–5.4 should be previous to the chapter 5.5. The language level also improved from the previous version. There are minor grammatical errors and most of them seem to be typos. However, these mistakes do not affect the general readability of the text.

4 Fulfilment of the assignment and applicability of the results

All aspects of the challenge have been addressed and covered in appropriate detail. The handicap of this work is that it is limited by time. Therefore, the result of the work probably cannot be used, as the malware itself stopped functioning in January 2015. However, as a tool for gaining practical experience it has a great value. There are good chances that the result will be used by me in courses on reverse engineering.

5 Activity and self-reliance of the student

The student showed high activity, initiative and independence. At the consultation meetings he was always well prepared.

6 Overall evaluation

It is unfortunate that the student didn't manage to break the encryption or identify the author of the virus. However, this topic is so challenging that even professionals teams in antivirus software companies sometimes fail. Regarding this, the student managed to deal with the assignment on a high level and meet the criteria requested. Considering all comments above, I recommend the thesis for defence with grade A.

Příloha B

Seznam použitých zkratek

FIT Fakulta informačních technologií ČVUT

kNN k Nearest Neighbours

API Application Programming Interface

CSV Comma-Separated Value

Příloha C

Obsah přiloženého CD

- `readme.txt` ..... stručný popis obsahu CD
- `jar` ..... adresář se spustitelnou formou implementace
- `src`
  - `impl` ..... zdrojové kódy implementace
  - `thesis` ..... zdrojová forma práce ve formátu LaTeX
- `text` ..... text práce
  - `thesis.pdf` ..... text práce ve formátu PDF