FAKULTA INFORMAČNÍCH TECHNOLOGIÍ ČVUT V PRAZE

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Název:	Souvislosti a doporučování obsahu na webu
Student:	Jindřich Žák
Vedoucí:	Ing. Lukáš Bařinka
Studijní program:	Informatika
Studijní obor:	Webové a softwarové inženýrství
Katedra:	Katedra softwarového inženýrství
Platnost zadání:	Do konce letního semestru 2019/20

Pokyny pro vypracování

Prostudujte informační architekturu webu FIT a sledované souvislosti jeho částí. Navrhněte způsob uložení potřebných metadat obsahu a jeho vzájemných vztahů pro web FIT do relační nebo grafové databáze s ohledem na prováděné výpočty nad metadaty. Navrhněte a naimplementujte aplikaci, která by umožňovala zadávání metadat a nad těmito metadaty pak navrhovala relevantní související obsah. Aplikaci otestujte jak z pohledu výstupů, tak z pohledu uživatelského rozhraní.

Seznam odborné literatury

Dodá vedoucí práce.

Ing. Michal Valenta, Ph.D.
vedoucí katedry

doc. RNDr. Ing. Marcel Jiřina, Ph.D.
děkan

V Praze dne 14. února 2019

FAKULTA
INFORMAČNÍCH
TECHNOLOGIÍ
ČVUT V PRAZE

Bakalářská práce

Souvislosti a doporučování obsahu na webu

Jindřich Žák

Katedra softwarového inženýrství
Vedoucí práce: Ing. Lukáš Bařinka

15. května 2019

Poděkování

Rád bych poděkoval vedoucímu mé bakalářské práce Ing. Lukáši Bařinkovi za jeho čas, cenné rady a připomínky, které mi během tvorby bakalářské práce poskytoval. Dále bych chtěl poděkovat Marii Böhmové za ochotu při konzultacích obsahové části webu. Také děkuji své rodině za podporu, kterou mi během studia poskytla, a za možnost studovat vysokou školu.

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, avšak pouze k nevýdělečným účelům. Toto oprávnění je časově, teritoriálně i množstevně neomezené.

V Praze dne 15. května 2019 ……

Odkaz na tuto práci

Žák, Jindřich. *Souvislosti a doporučování obsahu na webu*. Bakalářská práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2019.

Abstrakt

Práce se zabývá tvorbou doporučovacího systému pro web Fakulty informačních technologií Českého vysokého učení technického v Praze. Cílem práce je provést analýzu obsahu webu a na základě ní navrhnout a implementovat vhodné řešení pro hledání doporučovaného obsahu založeného na souvislostech mezi publikovanými příspěvky. Doporučování obsahu je dosaženo za pomoci uložení dat do grafové databáze Neo4j. Metriku, která je potřebná k ohodnocení míry souvislosti mezi články, poskytuje algoritmus PageRank. Web je založen na redakčním systému Drupal. Pro zajištění komunikace s grafovou databází byl vytvořen modul pro Drupal. Řešení práce má přínos pro návštěvníky, kterým zjednoduší orientaci na webu, a pro tvůrce obsahu, kterým ulehčí práci při úpravách obsahové části webu.

Klíčová slova doporučovací systém, analýza souvislostí, web FIT ČVUT, struktura webu, Neo4j, PageRank, PHP, Drupal

Abstract

This bachelor thesis describes a process of creation of a recommendation system for the web of Faculty of Information Technology, Czech Technical University in Prague. This thesis has two main goals: analyzing web content and proposing and implementing a suitable solution for recommending content based on associations between published articles. Content recommending is achieved by storing the data in the Neo4j graph database. The metric that is used for assessing the rate of association between articles is provided by the PageRank algorithm. The web is built on the Drupal content management system. To ensure communication with the graph database, a Drupal module was created. The solution makes navigating the website easier, therefore making it friendlier for page visitors. For administrators, the solution helps reduce the amount of work needed to maintain content.

Keywords recommending system, associations analysis, FIT CTU web, web structure, Neo4j, PageRank, PHP, Drupal

Obsah

Seznam obrázků

Seznam tabulek

Úvod

Web Fakulty informačních technologií slouží k předávání informací mezi fakultou a veřejností. Poskytuje informace věnující se různým tématům. Pro uživatele fakultního webu a jejich potřeby je nutné web uzpůsobit tak, aby jim jeho používání nepůsobilo žádné komplikace a uživatelská zkušenost byla co nejpříjemnější. Kromě přehledného uživatelského prostředí je důležité uživateli poskytnout vhodně organizovaná data pro jejich snadné procházení.

Cíl práce

Cílem této práce je zanalyzovat strukturu těchto dat a navrhnout vhodný způsob, jak uživateli nabízet relevantní související obsah, který je tématicky blízký k právě prohlíženému obsahu. Pro tyto účely bude nutné navrhnout, jak takový související obsah nalézat a jak jej uživateli nabízet.

Tato práce má hlavní přínos zejména pro návštěvníky webu, kterým bude usnadňovat navigaci napříč webem. Pomůže tak návštěvníkům nalézt informace, které hledají, rychleji. Zároveň ale tato práce pomůže tvůrcům obsahu, kterým ubyde práce s manuálním přiřazováním souvisejících článků.

Téma práce mne zaujalo, protože se věnuje tématu, které se dotýká přímo koncových uživatelů a klade si za úkol zjednodušit jim čas strávený na webu. Dále mne téma zaujalo tím, že propojuje web s moderními technologiemi pro ukládání dat.

Popis problému

Fakulta informačních technologií ČVUT v Praze připravuje v rámci modernizace nový fakultní web. Ten se bude skládat z příspěvků, které poskytují uživatelům informace. Jedním z hlavních požadavků na fakultní web je poskytovat uživateli přesné informace, které v dané chvíli potřebuje, a to ideálně

bez zdlouhavého procházení struktury webu. Vzhledem k vyššímu počtu příspěvků to může být pro uživatele obtížné.

Prvkem, který by k rychlému nalezení informací mohl pomoci, by mohl být krátký seznam souvisejících příspěvků na každé stránce. Ten by uživateli navrhoval, kde může získat další informace, které souvisí s aktuálně zobrazeným článkem. Pro takové účely bude nutné vyřešit problém, jak tyto související články získat.

V současné chvíli je tento problém řešen manuálním párováním souvisejících příspěvků v administraci webu. Toto řešení je ale pro tvůrce obsahu značně časově náročné a s rostoucím počtem příspěvků je navíc velmi obtížně udržitelné. Proto by bylo vhodnější, kdyby seznam souvisejících příspěvků poskytoval software.

Míra souvislosti mezi příspěvky závisí na shodných tématech, osobách, místech a dalších entitách, které mají vztah s daným příspěvkem. V databázi existuje velké množství vztahů mezi entitami. Tyto vztahy nesou informace, na základě kterých by mohlo být vhodné provádět analýzu souvislostí.

Struktura práce

V této práci se zabývám nejdříve analýzou daného problému. Do analýzy problému spadá seznámení se se současným stavem a strukturou dat. Analýza se dále zabývá možnostmi současných technologií pro ukládání dat, jelikož je reálné předpokládat, že k získání seznamu souvisejících příspěvků bude žádoucí ukládat zpracovávaná data ve vhodné podobě.

Kapitolou, která vychází z předešlé analýzy, je návrh řešení. Tato kapitola je věnována konkrétnímu návrhu architektury softwaru a procesu získávání doporučených příspěvků. Důležitou částí návrhu řešení je také navržení vhodného způsobu uložení dat, podle kterých bude software stanovovat míru souvislosti, s ohledem na rychlost a jednoduchost operací s nimi.

Další kapitola se zabývá samotnou implementací řešení. Tato kapitola popisuje průběh implementace a důležitá rozhodnutí, která bylo potřeba učinit. Samotná implementace vychází zejména z návrhu řešení, ve kterém jsou vyřešeny všechny důležité otázky týkající se implementace.

V průběhu implementace a po ní je důležité ověřit, zda implementovaný software funguje opravdu tak, jak bylo navrženo. Proto je jej nutné otestovat jednak z pohledu správnosti výstupů pomocí automatizovaných testů. Druhým způsobem, jak ověřit správnou funkčnost, je pomocí uživatelských testů. Pomocí nich je možné zjistit, zda doporučovaný související obsah je opravdu relevantní a usnadňuje uživatelům práci. Postupům aplikovaným při vylepšování výsledků a při testování je věnována kapitola Testování a ladění metriky.

Kapitola 1

Analýza

Pro kvalitní řešení problému je nejprve důležité se s daným problémem seznámit. Je třeba zjistit, jaký je současný stav a jakým způsobem je možné se z takového stavu dostat do situace, kdy je problém vyřešen. Současně s analýzou problému je vhodné prozkoumat, jaké existují možnosti pro realizaci takového řešení, jaké technologie bude možné využít. Těmto otázkám je věnována tato kapitola.

1.1 Analýza dat

Pro nalézání souvisejících příspěvků bude užitečné analyzovat příspěvky po jejich obsahové stránce. Samotné příspěvky obsahují jak samotná data, která jsou viditelná uživateli, tak metadata, která v sobě nesou důležité informace o příspěvku a nejsou pro návštěvníka viditelná. Jedná se například o data o souvislostech, data stanovující hierarchii příspěvků a podobně.

Pro navrhování souvisejících příspěvků je možné využít jak těchto metadat, tak i uživatelsky viditelných dat. U nich je však oproti metadatům často nutné počítat s nižší strojovou čitelností.

V databázi existují 4 typy entit. Jedná se o tyto typy:

- článek,
- osoba,
- událost,
- místo.

V následujících podkapitolách jsou rozebrána data, která se s entitami v databázi pojí. Podkapitoly jsou rozděleny dle zmíněných 4 typů entit.

1.1.1 Článek

Článek je základním typem obsahu fakultního webu. Předává uživateli především textové informace. Jeho použití je široké, záleží pouze na tvůrci obsahu, k jakému účelu článek využije. Články zároveň tvoří hlavní strukturu webu. Jsou organizovány hierarchicky. Každý článek si v sobě může nést odkaz na článek, který je jeho rodičem.

Tento vztah je přítomný u všech článků s výjimkou hlavní stránky. I ta je tvořena článkem. Jelikož je ale hlavní článek kořenem stromu, rodičovský článek je pro něj nepřiřazen. Další nepříliš důležitou výjimku tvoří stránky „Přístup odepřen“ a „Stránka nenalezena“. Rovněž se jedná o entity typu článek. Jelikož nejsou součástí informační struktury webu, oba tyto články stojí mimo hierarchii běžných článků bez přiřazeného rodičovského článku.

Nadpis

Základním prvkem každého článku je nadpis. Z důvodu, že nadpisy článků bývají mnohdy velice dlouhé, počítá fakultní web s použitím *krátkých titulků*. Ty jsou používány například v HTML tagu `<title>`. Na fakultním webu se tedy vyskytují nadpisy ve dvou formách – krátký a dlouhý nadpis. Pro příklad článku s dlouhým nadpisem „Katedra softwarového inženýrství“ je možné očekávat krátký nadpis „KSI“.

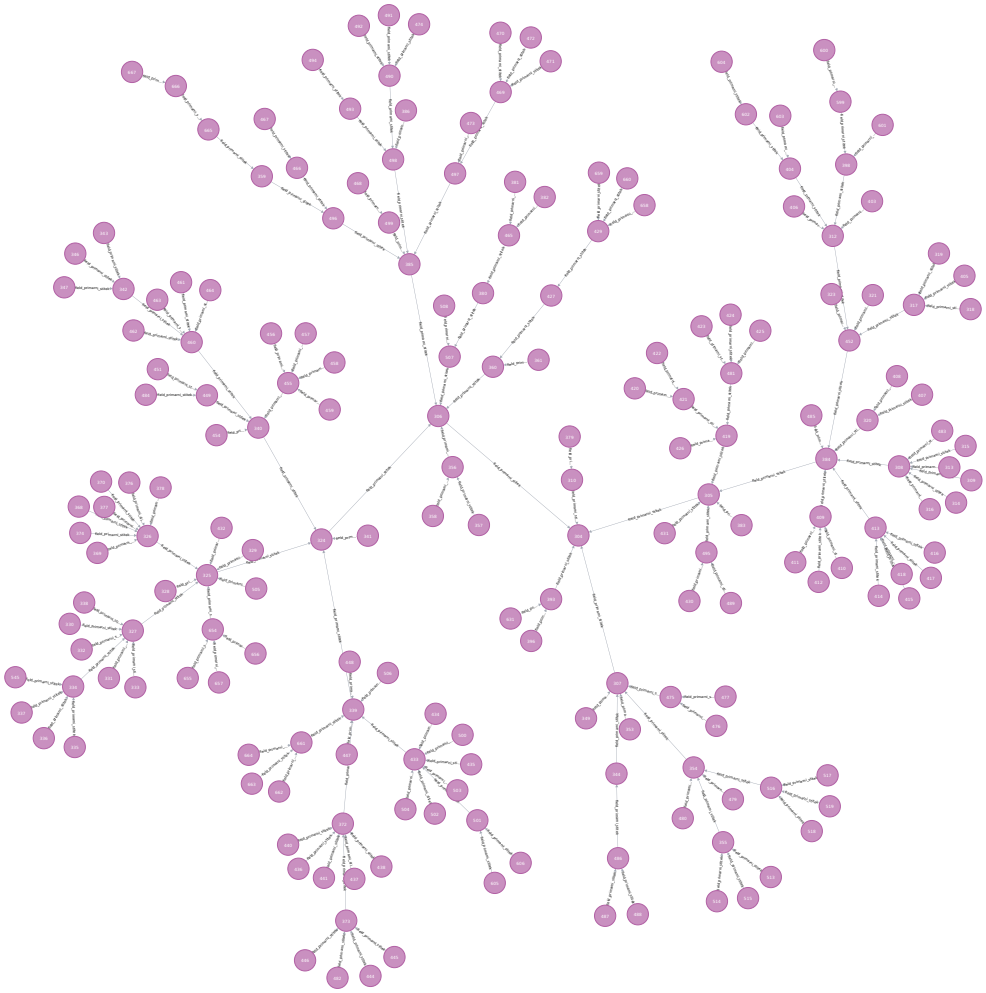
Při analýze dat článků je nutné počítat s výskytem různých pojmenování (aliasů) pro stejné objekty. Z těchto dvojic dlouhý – krátký nadpis lze získat mnoho takových aliasů. Díky nim je možné nalézat další vztahy mezi články.

Primární a sekundární štítky

Pro hierarchizaci článků počítá fakultní web s použitím štítků. Hlavní roli zde hraje takzvaný *primární štítek*. Jedná se o referenci na rodičovský článek. Takto spojené články tvoří pomocí referencí zakořeněný strom, jehož kořenem je titulní stránka webu. Je ovšem nutno dodat, že ze strany redakčního systému neexistuje omezení, které by vyřadilo možnost vytvořit pomocí takových vztahů cyklus. Na obrázku 1.1 je zobrazena vizualizace stromu tvořícího hlavní hierarchii.

Hierarchická struktura umožňuje uživateli postupný průchod do nižších hladin stromu. Je předpokládáno, že uživatel při průchodu navštěvuje nejdříve články, které jsou v hierarchii blízko kořene. Takové články obsahují většinou pouze základní informace a poskytují uživateli souhrnný obecný přehled o nějakém tématu.

Takové široké téma je po menších částech rozebráno v potomcích daného článku. Jak se návštěvník postupně dostává k nižším hladinám stromu, dostává se stále více ke konkrétním informacím. V nižších hladinách stromu získává návštěvník z článků podrobnější a úžeji zaměřené informace.



Obrázek 1.1: Vizualizace stromu článků tvořícího hlavní hierarchii

Z hierarchie článků lze získat užitečné informace pro hledání souvisejících článků. Při návštěvě nějakého článku, který má přiřazeného svého rodiče, lze uvažovat, že informace z jeho předchůdců jsou uživateli již z určité části známé. S určitou pravděpodobností se totiž uživatel dostal na současný článek právě přes jeho předchůdce. Takto je možné pokusit se zpětně určit informace o uživatelově cestě webem, aniž by byla na serveru zachovávána data o zobrazených článcích daným uživatelem.

Pro související návrhy by tedy měly být upřednostněny spíše články, které jsou nejbližšími potomky současného článku. Právě ty budou na aktuální článek informačně navazovat nejvíce. Mezi navrhovanými články by se neměly často zobrazovat předchůdci aktuálního článku. Mimo jiné proto, že na rodiče aktuálně zobrazeného článku se může uživatel dostat velice rychle pomocí

drobečkové navigace, kterou nový fakultní web používá.

Další užitečnou informací, která je s článkem svázána, jsou *sekundární štítky*. Jejich princip je stejný jako u *primárních štítků* – tedy zapojení článku do stromové struktury. Jejich použitím je docíleno, že článek může být součástí více než jedné (hlavní) hierarchie. Uživateli je však zobrazována pouze hierarchie tvořená použitím *primárních štítků*. Vztahy v sekundárních hierarchiích můžou mít uplatnění při analýze souvislostí mezi články podobně jako v případě primární hierarchie.

Doporučený obsah

Struktura článku umožňuje ručně zadávat takzvaný *doporučený obsah*. Jedná se o seznam referencí na další články. V současné chvíli je to jediný způsob, jak uživateli nabízet související obsah. Jde o způsob, který je však pro tvůrce obsahu časově náročný. Systém, který je předmětem této práce, by měl z velké části zastoupit tento manuální způsob zaznamenávání souvisejících článků.

Softwarovou cestou ale nebude vždy plně možné nalézt všechny souvislosti. Není totiž zaručeno, že na každém článku budou zadána všechna relevantní data a metadata, na základě kterých by bylo možné souvislosti nalézat. Proto není ideálním řešením manuální způsob nalézání souvislostí kompletně nahradit softwarovým řešením. Naopak je vhodné využít výhody obou způsobů tak, aby se vzájemně doplňovaly.

Předcházející – následující článek, složený článek, souvislosti

Web umožňuje vytvoření série článků pro případ, kdy se bude stejnému tématu věnovat více článků, které na sebe budou postupně navazovat. Typicky to platí například pro články věnující se různým ročníkům nějaké konference či sériím přednášek. Pro takový případ je možné k článku přiřadit *předcházející* a *následující článek*.

Některé články jsou odvozeny od jejich rodičovského článku. Rodičovský článek se může věnovat například nějaké události. Příkladem potomka, který je od takového rodiče odvozený, by mohl být konkrétní ročník dané události. Takový článek bude na webu označen příznakem *složený*. Je pravděpodobné, že uživatel bude chtít z rodičovského článku navštívit právě potomka, který je složeným článkem. Na základě tohoto zjištění je možné očekávat mezi takovými dvěma články zvýšenou míru souvislosti.

Mezi entitami dále existuje vztah typu *souvislosti*. Tento vztah slouží k zachycení věcných souvislostí, které mezi entitami vznikají. Jedná se zejména o souvislosti, které nemohou být zaznamenány pomocí jiných konkrétněji specifikovaných vztahů.

Kontakt, zodpovědná osoba

V databázi se nachází velké množství osob, které na fakultě působí, nebo s ní jsou spojeny. Tyto osoby jsou často spojeny s konkrétními články pomocí vztahů *kontakt* a *zodpovědná osoba*. Vztah typu *kontakt* nese informaci o tom, s kým je téma daného článku spojeno, koho může návštěvník kontaktovat, v případě, že si přeje získat bližší informace. Vztah *zodpovědná osoba* přiřazuje článku osobu, která ručí za jeho obsahovou správnost.

Z hlediska souvislostí hraje důležitější roli vztah typu *kontakt*, jelikož propojuje dvě entity (článek s osobou), které spolu významově souvisí. Oproti tomu u vztahu *zodpovědná osoba* existuje větší pravděpodobnost, že takový vztah bude existovat mezi článkem a osobou, která pouze ručí za obsahovou správnost, ale s tématem není nijak blízce spojena.

1.1.2 Osoba

Databáze fakultního webu umožňuje dále kromě článků vytváření a ukládání osob. Osobou v rámci fakultního webu může být kdokoliv, kdo je spojen s Fakultou informačních technologií. Entita je tvořena základními informacemi o osobě jako je její jméno, příjmení, email či pracovní zařazení na fakultě.

Osoba bývá nejčastěji spojena s článkem pomocí vztahu *kontakt* či *zodpovědná osoba*. Osoba ale může být ve vztahu s článkem například na základě vztahu typu *souvislost*. Jelikož tyto a další možné vztahy jsou již popsány v podkapitole 1.1.1, není je zde třeba znovu detailně popisovat.

1.1.3 Událost

Událost je dalším typem entit, které je možné vytvářet na webu fakulty. Událost se věnuje akcím či událostem, se kterými je Fakulta informačních technologií nějak spojena. Může se jednat o přednášky, dny otevřených dveří či například kulturní akce pořádané studenty.

Časové a s událostmi spojené údaje

U příspěvků, které se věnují akcím či událostem, je možné zadat i interval data konání. Jelikož aktuálnost takových příspěvků se v čase mění, tuto informaci lze využít pro změny míry souvislosti v různých časových okamžicích. Například by nedávalo smysl navrhovat ke čtení příspěvek, který se věnuje akci pořádané až za mnoho měsíců či naopak akci již dávno uplynulé.

S blížícím se datem konání bude relevance příspěvku pro uživatele naopak růst. Relevance takového příspěvku bude pro návštěvníka dosahovat nejvyšších hodnot od okamžiku, kdy budou do data konání události zbývat řádově jednotky dní. Několik dní po konci události relevance pro uživatele opět postupně klesne.

Vztah s článkem a místem konání

Jelikož entita typu *událost* neobsahuje mnoho obsahu, nezobrazuje se na samostatné stránce ale pouze jako součást stránky jiné. Pro propojení s *článkem*, který se blíže věnuje dané *události*, existuje v databázi vztah jménem *článek*. Tento vztah slouží k propojení entity typu *událost* s příslušným *článkem*.

U příspěvků popisujících události je dále možné zadat místo konání. Tato informace je pro nalézání souvislostí rovněž užitečná, protože umožňuje seskupování příspěvků na základě místa. Události je možné skrýt po jejich skončení, pokud se nejedná o důležitou událost, která má svůj význam i po jejím skončení.

1.1.4 Místo

Posledním typem entity, který je možné na fakultním webu vytvořit, je *místo*. Popisuje místo, které se pojí často s událostmi či samotnými osobami. Taková entita má v sobě uložené pouze základní informace o místě jako jeho název, údaj, v které budově se místo nachází, či vybavení, které v daném místě je.

Veškeré entity včetně jejich vztahů jsou zobrazeny na obrázku 1.2.

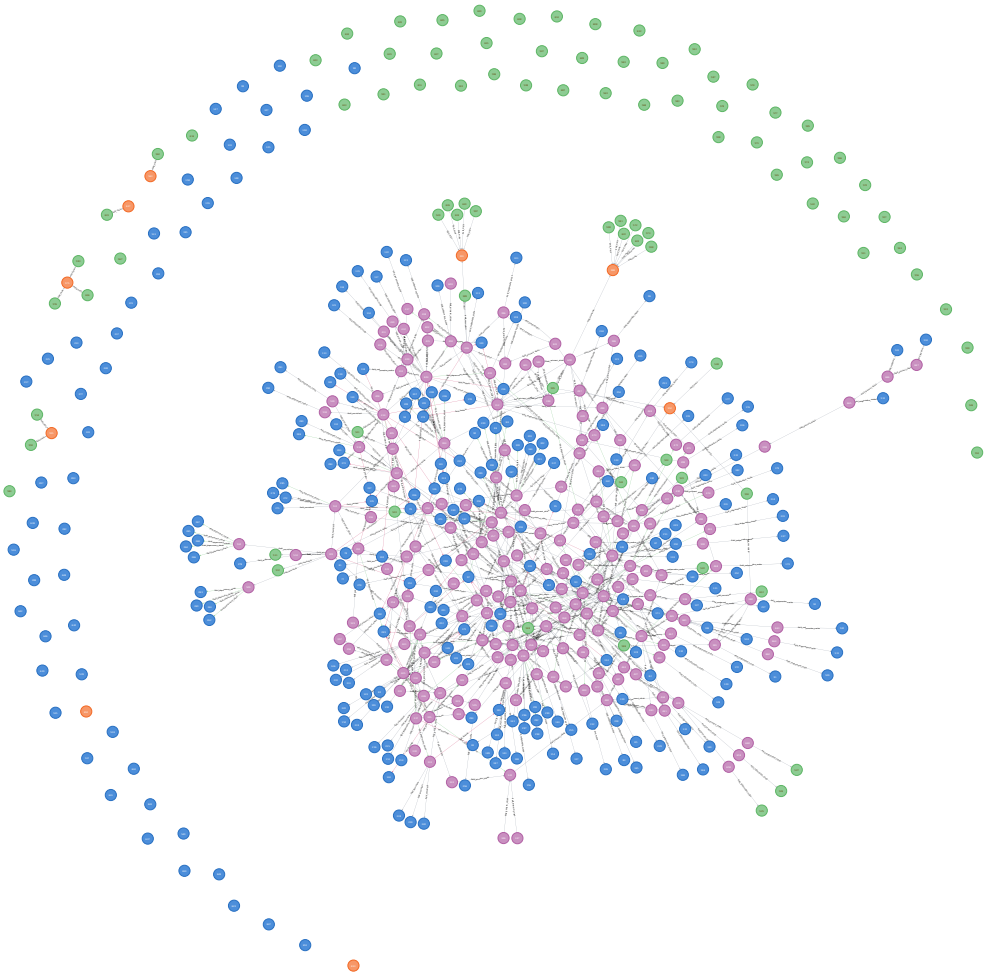
1.2 Analýza uživatelů a jejich potřeb

Přívětivost fakultního webu je pro uživatele důležitým aspektem. Uživatele lze dělit na dvě skupiny podle jejich role vůči webu. První skupinou jsou návštěvníci, kteří mají přístup pouze k veřejné části webu a hledají na ní informace o fakultě. Druhou skupinou jsou tvůrci obsahu, kteří mají přístup do webové administrace a zodpovídají za tvorbu publikovaného obsahu.

1.2.1 Návštěvníci webu

Pro nabízení co nejvhodnějšího obsahu návštěvníkovi je potřeba znát jeho preference a vědět, jaká témata ho zajímají, jaké informace hledá. Zdroje informací, na základě kterých lze preference určit, jsou ale do jisté míry omezené. Je důležité zmínit, že na webu neexistuje a není plánováno přihlašování návštěvníků, díky kterému by se mohly jejich preference zaznamenávat a vázat k danému účtu.

Takové řešení by pro tento případ nebylo ani příliš vhodné. Lze totiž uvažovat, že uložené informace by byly značně zkreslující, protože každá návštěva stejného návštěvníka na fakultním webu může mít odlišný důvod. To, jaké informace návštěvník hledá, se s časem mění. Hlavním zdrojem informací, který je možné použít, je jeho aktuální průchod webem.



Obrázek 1.2: Vizualizace grafu příspěvků uložených v databázi fakultního webu

Klasifikace návštěvníků

Fakultní web navštěvují uživatelé, mezi kterými lze vypozorovat podobnosti v jejich vztahu vůči fakultě. Na základě těchto podobností je možné sestavit seznam skupin, do kterých půjde zařadit většinu uživatelů. Určení preferencí uživatelů lze zpřesnit zařazením uživatele do jedné z takových skupin. Na základě vyhledávané informace je možné s určitou pravděpodobností určit příslušnost daného uživatele k některé ze skupin.

Zde je seznam skupin, které pokrývají většinu uživatelů webu:

- zájemce o bakalářské studium,
- student bakalářského programu,
- zájemce o magisterské studium,
- student magisterského programu,
- zájemce o doktorské studium,
- student doktorského programu,
- zaměstnanec,
- partnerské firmy a organizace,
- média,
- široká veřejnost.

V konkrétním případě je tedy možné se pokusit stanovit příslušnost ke skupině na základě zobrazených příspěvků daným návštěvníkem. Je ovšem nutné zmínit, že nikdy nelze se stoprocentní jistotou určit, do které skupiny návštěvník patří. Je například možné, že určitý návštěvník se bude po webu pohybovat „chaoticky“ a bude navštěvovat články, které jsou určeny pokaždé primárně pro jinou skupinu. V takovém případě je zbytečné se pokoušet stanovit příslušnost ke skupině, naopak by to mohlo působit ku neprospěchu věci.

Také se může stát, že někteří návštěvníci budou patřit do více skupin. Například student bakalářského programu bude s nemalou pravděpodobností zároveň zájemcem o magisterské studium. Tento blízký vztah mezi skupinami hraje roli při hledání souvisejících příspěvků. Mezi skupinami může ale zároveň existovat i výlučný vztah. Například případ, ve kterém figuruje návštěvník zároveň jako zájemce o bakalářské studium a magisterské studium, není příliš reálný.

Navzdory situaci, kdy bude příslušnost návštěvníka k určité skupině (či skupinám) stanovena s vysokou pravděpodobností, stále nelze vyloučit, že takový návštěvník bude mít zájem i o příspěvek určený pro relativně odlišnou

skupinu. Žádný příspěvek by proto neměl být vyřazen z doporučování na základě příslušnosti návštěvníka k některé ze skupin. Příslušnost návštěvníka ke skupině může pouze posílit nebo naopak zeslabit pravděpodobnost doporučení příspěvků. Každý příspěvek tedy musí mít vždy nenulovou pravděpodobnost, že se dostane do seznamu doporučených příspěvků jakémukoliv návštěvníkovi.

Realizace klasifikace návštěvníků

Nejjednodušším způsobem je získávat informace o návštěvníkovi na základě aktuálně zobrazeného příspěvku. Web je organizován do stromové hierarchie. U mnoha příspěvků platí, že jejich potomci se budou s velkou pravděpodobností věnovat podobným tématům, a tedy že tyto příspěvky budou zároveň určeny návštěvníkům s blízkými preferencemi.

Například příspěvek „Bakalářský studijní program“ bude primárně určen pro skupinu *student bakalářského programu*, příspěvek „O fakultě“ bude naopak cílit na návštěvníky, kteří hledají obecné informace a základní údaje o fakultě (*zájemci o studium*, *široká veřejnost*). Vzhledem ke stromové struktuře lze říci, že všichni potomci daného příspěvku jsou pravděpodobně určeni pro stejnou skupinu.

Příspěvky mají v sobě uložená různá data. Jde zejména o data textová, časová, dále se na článcích objevují různé příznaky nebo data znázorňující vztah mezi entitami. S využitím takových dat je opět možné pokusit se blíže stanovit, jaké jsou návštěvníkovy preference.

Kromě navrhování souvisejících příspěvků na základě aktuálního příspěvku je možné analyzovat i celou návštěvníkovu cestu webem. Z cesty lze získat data o tom, jak se vyvíjí návštěvníkovy preference obsahu, jaká témata ho postupně zaujala. Na základě takových informací může být možné pokusit se predikovat, kam se bude návštěvník v rámci webu ubírat dále.

Díky znalosti celého průchodu webem lze dále vyřadit příspěvky, které již byly navštíveny. Je sice možné, že návštěvník se bude chtít vrátit na příspěvky, které již navštívil. Cestu, jak se na takové příspěvky vrátit, ale návštěvník již zná, nebo ji má uloženou v historii webového prohlížeče. Proto není příliš vhodné zobrazovat doporučení na navštívený příspěvek, jelikož by návštěvníkovi orientaci na webu příliš nezjednodušil.

1.2.2 Tvůrci obsahu

S hlavním úkolem tvůrců, kterým je vytvářet nový obsah, souvisí i úkol udržovat existující obsah aktuálním. Vzhledem k rozsáhlé působnosti fakulty existují na webu stovky příspěvků věnující se různým tématům, které je třeba udržovat v aktuálním stavu.

Jak již bylo nastíněno, v příspěvku je uloženo množství dat. Jejich množství napomáhá tomu, že je možné příspěvky analyzovat dle různých požadavků. Zároveň ale tvůrce obsahu staví do situace, kdy musejí toto velké

množství dat spravovat a kontrolovat jeho aktuálnost. S narůstajícím počtem příspěvků je tento stav hůře a hůře udržitelný.

Z velké části je tento problém týkající se kontroly aktuálnosti dat řešitelný pouze pomocí člověka. U mnoha dat platí, že počítač si zkrátka není vědom informací, kterých si člověk vědom je. Existují ale i data, která vycházejí z velké části z dat jiných. Pokud taková data dokáže na základě jiných dat počítač vytvořit, není u nich přítomnost člověka nutná.

Mezi články se například vyskytuje vztah *doporučený obsah*. Tento vztah popisuje, které příspěvky budou návštěvníkovi nabízeny, když si na webu zobrazí určitý příspěvek. A právě tento vztah do značné míry závisí na ostatních datech příspěvku. Vyšší míra shody dat mezi dvěma příspěvky může právě napovídat, že mezi těmito příspěvky bude vztah *doporučený obsah* existovat.

1.3 Analýza procesů a možných případů užití

Pro správné řešení problému doporučování příspěvků je kromě analýzy uživatelů zároveň nezbytné pochopit i procesy, ve kterých uživatelé figurují. Těmto otázkám je věnována následující kapitola.

1.3.1 Návštěvníci webu

Návštěvníci webu jakožto konzumenti obsahu přicházejí na web z důvodu získání informací. Dle jejich klasifikace z předešlé kapitoly je možné dále specifikovat, za jakým účelem informace hledají. Může se jednat například o získání informací za účelem ucházení se o studium, hledání studijních informací či získání tiskových zpráv pro média.

Navigační prvky

K procházení webu využívají návštěvníci navigačních prvků na stránce, které jim poskytují možnosti, kam se dále v rámci webu ubírat.

V současné chvíli na fakultním webu existuje několik jednoduchých navigačních prvků. Jedná se o nabídku s příspěvky, které jsou v hlavní hierarchii potomkem aktuálního příspěvku. Dále jde o drobečkovou navigaci, která naopak zobrazuje cestu z kořenu stromu k aktuálnímu příspěvku.

Všechny doposud zmíněné prvky čerpají pouze z hierarchické organizace příspěvků. Kvůli tomu může docházet k situacím, kdy jakmile návštěvník vstoupí do určitého podstromu z hierarchie příspěvků, je pro něj poměrně obtížné dostat se do podstromu jiného. Pomoci zamezit takovému scénáři může právě prvek s doporučenými příspěvky, který návštěvníkovi poskytne alternativy, kam se může v rámci webu dále ubírat, aniž by zcela změnil téma příspěvků.

Kromě navigačních prvků, které čerpají z hierarchické organizace článků, na webu existují i prvky, které zobrazují odkazy na články na základě ne-

hierarchických dat. V postranním panelu jsou zobrazeny například seznamy článků, které jsou s aktuálním článkem ve vztahu typu *souvislost*, externí odkazy, odkazy v rámci fakultního webu či kontakty na osoby spojené s aktuálně zobrazeným článkem.

Pro tuto práci je ovšem nejdůležitější seznam s doporučenými články. Zobrazuje seznam článků, které byly pro aktuálně zobrazený článek manuálně přiřazeny pomocí vztahu *doporučený obsah*. Prvek s doporučenými příspěvky je určen pro návštěvníky s cílem zjednodušit jejich orientaci na webu a zpříjemnit uživatelskou zkušenost.

Průchod webem

Průchod webem je posloupnost příspěvků navštívených v rámci jedné návštěvy webu. Pro analýzu cesty webem je důležité určit, na jaké stránce uživatelova návštěva začala. Lze uvažovat, že ve velké části případů se bude jednat o hlavní stránku webu.

V takovém případě jsou pomocí hlavní nabídky uživateli nabízeny potomci hlavní stránky v hierarchii článků. Je předpokládáno, že s časem stráveným na fakultním webu se bude návštěvník postupně ubírat ke stále více konkrétním a úzce zaměřeným článkům. Pokud by se uživatel dostal až k listu celé hierarchie a stále nenalezl hledané informace, dostává se do situace, kdy nové informace může získat pouze v jiných podstromech hierarchie. V takové chvíli musí využít nehierarchických navigačních prvků – například seznamu doporučených článků.

Seznam doporučených článků získává na důležitosti v několika případech. Prvním případem je zmíněná situace, kdy se návštěvník neúmyslně ocitne v nesprávném podstromu. Zde mu seznam doporučených článků může poskytnout cestu do jiného podstromu bez velké změny v tématech.

Seznam doporučených článků může mít silné využití i v případě, kdy návštěvník vstoupí na web bez přesného cíle, kterého chce dosáhnout. Takový uživatel prochází webem bez definovaného cíle, jde mu spíše o zjištění zajímavých informací o fakultě, o utvoření si obecnějšího obrazu o ní. Seznam doporučených článků může takovému návštěvníkovi pomoci „rozšířit obzory“ o fakultě.

Jelikož každá návštěva webu znamená vznik nového průchodu webem, je možné, že z takto vzniklých průchodů se budou některé průchody velmi podobat, či budou totožné. V případě existence průchodu webem, který je mezi návštěvami uživatelů často vyskytuje, může opět pomoci seznam doporučených článků. Ten může v takovém případě poskytnout návštěvníkům jakousi „zkratku“, která uživatele dovede k hledané informaci rovnou a ušetří mu tak čas při absolvování celé cesty.

Příchod z vyhledávače

Specifická situace nastává v případě, kdy návštěvník na web vstoupí z odkazu z internetového vyhledávače. Takový případ lze detekovat pomocí hodnoty `referer` v HTTP hlavičce. [1] Jestliže návštěvník ve vyhledávači formuloval přesný dotaz a skrze výsledky se dostal na fakultní web, lze uvažovat, že velkou část práce při vyhledávání informace odvedl již samotný vyhledávač. Ten návštěvníka pravděpodobně nasměroval na správný či velice blízký příspěvek.

Pro takový případ by bylo vhodné uživateli doporučovat pouze příspěvky věnující se stejným či velice blízkým tématům. Je reálné uvažovat, že uživatel si v takové situaci nepřeje zobrazovat doporučení příspěvků, které mu „rozšíří obzory“ ohledně fakulty. Uživatel, který již formuloval jasný dotaz ve vyhledávači, hledá nejspíše jasné informace.

1.3.2 Tvůrci obsahu

Ačkoliv je tato práce primárně věnována usnadnění orientace návštěvníků na fakultním webu, má dopad i na tvůrce obsahu. Tvůrce obsahu je člověk, který má přístup do administrace webu a smí obsah vytvářet, upravovat nebo mazat.

Databáze fakultního webu umožňuje v příspěvku vyplnit množství informací. To umožňuje s daty dále pracovat a strojově je zpracovávat. Nevýhodou je, že vyplňování těchto dat a udržování je aktuálními je pro tvůrce časově náročný úkol.

V databázi v současnosti existuje okolo dvou set článků. Aby měl každý článek přiřazených alespoň pět doporučených článků, by bylo nutné ručně vytvořit až tisíc vazeb typu doporučený obsah. V případě, kdy v databázi tvůrce obsahu přidá nový příspěvek, musí navíc najít, pro které články bude nově vzniklý článek doporučený. Tento úkol je rovněž časově náročný.

Dalším problémem je, že aby byl tvůrce webu schopen stanovit pro daný příspěvek doporučený obsah, musí se velice dobře orientovat v článcích na fakultním webu a mít o nich celkový přehled. Jelikož počítač může oproti člověku během okamžiku projít všechny články v databázi, není to pro něj problém. Automatické doporučování článků by pro tvůrce obsahu bylo ulehčením práce, které by jim umožnilo věnovat více času úkonům, s kterými jim počítač nemůže pomoci.

Existuje zde ovšem riziko, že v databázi webu neexistuje dostatečné množství informací, na základě kterých lze s vysokou jistotou určovat související články. Některé informace, kterých si je tvůrce obsahu vědom, naopak nemusí být možné zaznamenat do databáze webu. Tvůrce webu převyšuje počítač svou intuicí, pochopením významu příspěvku. Proto by tvůrci obsahu neměli být z procesu určování doporučených příspěvků plně vyřazeni.

Možné využití automatické analýzy příspěvků

Nad daty zpracovávanými pro doporučování příspěvků by dále bylo možné provádět další operace, které sice nejsou součástí rozsahu této práce, ovšem stojí za zmínku z důvodu případné rozšiřitelnosti systému do budoucna.

Detekce chyb v hierarchii V rámci analýzy souvislostí ve vzniklém grafu by bylo možné detekovat, zda by některý příspěvek neměl patřit v hierarchii na jinou pozici, než na které se nachází. Takovou situaci může indikovat například zvýšené množství vztahů s příspěvky, které se nacházejí v rozdílném podstromu grafu. Takové upozornění ze strany doporučovacího systému by tvůrcům obsahu pomohlo při případné reorganizaci hierarchie příspěvků.

Nekonzistence dat příspěvku Dalším případem, ve kterém by mohl mít doporučovací systém přínos pro tvůrce obsahu, je hledání nekonzistence v datech. Nekonzistence dat může nastat například v případě, kdy příspěvek uvedený v seznamu doporučených příspěvků nebyl návštěvníky skrze toto doporučení často navštěvován. Takový stav by mohl indikovat, že metadata použitá k tvorbě doporučení nekorespondují s vlastním obsahem příspěvku a příspěvek je tedy doporučován tam, kde by doporučen být neměl. Takový případ by následně mohl být předložen tvůrci obsahu na kontrolu.

Získání veškerých souvislostí Pro tvůrce obsahu by dále bylo užitečné získat kompletní seznam příspěvků, které jsou spojeny s určitou osobou, událostí nebo místem. Potřeba takového seznamu nastává v případě, kde dochází ke změnám informací u nějaké entity (například když osoba ukončí spolupráci s fakultou). Jelikož existuje provázanost informací mezi články a ostatními entitami, je v případě změny dat u některé entity nutné provést kontrolu aktuálnosti dat i v článcích spojených s touto entitou. V takové situaci by mohl doporučovací systém poskytnout všechny související příspěvky pro jejich kontrolu a případnou aktualizaci obsahu.

Detekce zastaralých příspěvků V rámci analýzy dat bylo zjištěno, že na současném fakultním webu existují příspěvky, které již dlouhou dobu nejsou aktuální a k jejich nalezení stačí pouze pár kliknutí z titulní stránky webu (například dávno ukončené soutěže). Na takové příspěvky by tvůrce obsahu opět mohl být upozorněn při procházení grafu příspěvků.

1.4 Analýza technologií

Jelikož se u doporučovacího systému jedná z velké části o technické řešení, tato kapitola se bude věnovat právě analýze současného stavu systému, na kterém

fakultní web běží a také technologickým možnostem, které budou využitelné při tvorbě doporučovacího systému.

1.4.1 Současný stav

Nově vznikající fakultní web je postaven na redakčním systému Drupal ve verzi 8. Ten běží na webovém serveru Apache. Drupal je open source redakční systém, který bývá někdy označován také za redakční framework (content management framework). [2] Je to z toho důvodu, že ačkoliv je Drupal v základní verzi použitelný bez zkušenosti s programováním, je zároveň vysoce modifikovatelný pro specifické potřeby uživatelů. Modifikace Drupalu je možné realizovat pomocí modulů.

Data jsou uložena v relační databázi MySQL. Drupal nahlíží na uložený obsah jako na uzly (nodes). Ať už se jedná o článek či komentář, z pohledu Drupalu se jedná vždy o uzly, které v sobě nesou strukturovaná data. V administraci Drupalu je možné vytvářet nové typy obsahu (content types), které definují strukturu dat nově vytvořených uzlů. V případě fakultního webu je možné mluvit o čtyřech typech obsahu – *článek*, *událost*, *osoba*, *místo*. Tyto typy obsahu jsou blíže popsány v kapitole 1.1.

Dle pohledu Drupalu je uzel složen z dat a metadat. [2] Dle Drupalu jsou data veškeré informace, které jsou zobrazeny návštěvníkovi. Metadata jsou oproti nim skryta a slouží pro interní účely systému jako například určení, kdy a jak má být uzel zobrazen.

Základní funkcionalita Drupalu je obsažena v takzvaném Drupal Core. Jedná se o základ potřebný ke spuštění Drupalu. Skládá se z modulů, které poskytují základní funkcionalitu. Pokud tato funkcionalita nepostačuje specifickým potřebám uživatele, je možné doinstalovat modul vytvořený třetí stranou. Mnoho takových modulů je volně ke stažení ve veřejném repozitáři. [1]

Moduly v architektuře Drupalu stojí nad datovou vrstvou, která je v struktuře vrstev postavena nejníže. Data z ní získaná jsou poskytnuta výše vrstvě modulů, ve které je možné vykonávat logiku s nimi spojenou. V další vrstvě dochází k sestavení takzvaných *bloků* (blocks). *Bloky* jsou elementy, ze kterých se skládá výsledná stránka vykreslená uživateli. Samostatným *blokem* může být například seznam s doporučenými články. Ten může být administrátorem webu libovolně umístěn na jakékoliv místo na stránce.

Ne každý uživatel musí mít v rámci webu stejná práva, proto další v pořadí je vrstva, která rozhoduje o tom, která data je možné danému uživateli zpřístupnit. Poslední vrstvou je prezentační vrstva, která uživateli zobrazuje daný obsah pomocí šablonovacího systému Twig.

Důležitou funkcionalitu, kterou Drupal poskytuje, je možnost provádět automatické opakované procedury pomocí softwarového daemonu CRON. Drupal tyto opakované procedury využívá například k pravidelné kontrole aktua-

[1] `https://www.drupal.org/project/project_module`

lizací. Tyto procedury mohou být spuštěny koncovými uživateli při návštěvě stránek nebo s použitím příkazu `cron` či obdobných nástrojů (např. *Scheduled Task* ve Windows). CRON je možné konfigurovat v administraci Drupalu.

1.4.2 Technologie pro analýzu chování uživatele

Pro personalizovaný doporučovací systém je nutné znát preference uživatele. Ty lze získat nejen z aktuálně zobrazeného příspěvku ale i z jeho celkové cesty webem a jeho chování na stránkách. Technologickým možnostem, které toto umožňují, je věnována tato podkapitola.

Mezi řešení, jak monitorovat uživatelovu cestu webem, patří služba Google Analytics. Pro komunikaci s Google Analytics je možné využít Real Time Reporting API, které umožňuje získávání dat o provozu na webových stránkách v reálném čase. [3]

Další alternativou je monitorovat uživatelovu cestu a chování na webu je vytvoření vlastního systému, který by tato data byl schopný zpracovat. Aby bylo možné sestavit uživatelovu cestu, je nutné vyřešit identifikaci uživatele.

Identifikaci uživatele je možné realizovat pomocí ukládání cookies na uživatelově zařízení, podobně jako to dělá služba Google Analytics. Alternativou k použití cookies pro identifikaci může být předávání unikátních query parametrů v adrese URL nebo analýza hodnoty `referer` v HTTP hlavičce. [1]

Další možností je použití JavaScriptové knihovny (např. Fingerprint.js) pro zjištění unikátního „otisku prstu“ zařízení na základě jeho hardwarové a softwarové konfigurace, kterou je možné pomocí JavaScriptu získat. Tato možnost ovšem nemusí mít stoprocentní úspěšnost, obzvlášť na mobilních zařízeních. [4]

1.4.3 Technologie pro uchování dat

Pro efektivní zpracování dat je vhodné zabývat se možnostmi uchování dat. Této problematice se věnuje tato kapitola.

Relační databáze

Relační databáze umožňují uložení jednoduše strukturovaných dat do předem definovaných tabulek. Relační databáze se hodí pro použití, kdy struktura dat, které je třeba uchovat, má tabulkovou povahu. V případě relační databáze je kladen důraz hlavně na data v samotných entitách než na vztahy mezi nimi. Ačkoliv relační databáze je schopna uchovat relace mezi entitami, nevypořádávají se s nimi optimálně. [5]

Vztahy mezi entitami v relační databázi jsou realizovány ukládáním cizích klíčů do entit. To vede k tomu, že k procházení vztahů je nutné použít spojení (*JOIN*). Takové operace jsou výkonově náročné a jejich časté užívání může vést ke snížení výkonu výsledné aplikace. [6] Další komplikaci přinášejí *many-to-many* vztahy, které vyžadují vytvoření speciální vazební tabulky.

Grafové databáze

Grafové databáze jsou oproti těm relačním lépe přizpůsobeny na uchovávání dat, která jsou vysoce propojená a existuje mezi nimi velké množství vazeb, které je nutné procházet. Struktura grafové databáze není tvořena tabulkami podobně jako relační databáze, ale grafem tvořeným vrcholy a hranami. Díky tomu často poskytují možnost zachytit cílovou doménu lépe a přehledněji. Nad takovým grafem je možné provádět operace od jednoduchého hledání vrcholů po komplexní grafové algoritmy.

Klíčovou vlastností grafových databází je takzvaná *index free adjacency*, díky které je rychlost procházení grafu zásadně rychlejší oproti relačním databázím. Tato vlastnost určuje, že každý vrchol odkazuje přímo na sousední vrcholy. [5] Díky tomu není nutné se spoléhat na procházení indexů podobně jako v případě relační databáze.

Nejčastěji používaným grafovým modelem je tzv. *labeled property graph*. [5] Takový model je tvořen vrcholy a vztahy mezi nimi, přičemž platí, že každý vrchol může mít své pojmenování (angl. *label*) užitečné pro kategorizaci a následné dotazování (například „osoba“, „město“, …). Zároveň mohou být ve vrcholu uložena data vlastností ve formě klíč – hodnota. Vztahy jsou realizovány orientovanými hranami mezi vrcholy a rovněž mohou mít své vlastnosti a pojmenování (například „je přítelem“, „bydlí v“, …). Takové schéma je v porovnání se schématem relační databáze flexibilnější vůči změnám, jelikož nevyžaduje úpravu tabulek. [7]

V závislosti na problémové doméně může však použití relační databáze vyústit v jednodušší databázové schéma. Tento případ může nastat zejména, když problémová doména nevyžaduje použití mnoha vztahů a primární důležitost mají entity samotné. Dále je nutné dodat, že grafová databáze s sebou přináší nutnost použití NoSQL dotazovacích jazyků, což může být zejména v jednoduchých projektech nezanedbatelná komplikace.

Neo4j Neo4j je open source grafová databáze. K dotazování se nad daty používá dotazovací jazyk Cypher. Pro rozšíření funkcionality pro práci s daty umožňuje doinstalování pluginů. Tímto způsobem je možné přidat možnost spouštět nad daty grafu například grafové algoritmy. K Neo4j existuje podrobná dokumentace včetně článků věnujících se jejímu použití.

OrientDB Jednou z alternativ je databázový engine OrientDB. Jedná se o takzvanou *multi-model* databázi. To znamená, že kombinuje výhody dokumentové databáze s databází grafovou. Díky tomu je schopna obsáhnout širší spektrum možných využití.

Ostatní typy databázových systémů

Kromě relačních a grafových databází existují dále databáze dokumentové. Jejich základním rozdílem oproti relačním databázím je absence předem definovaného schématu, proto se hodí pro použití v doménách, kde jsou předpokládány časté změny schématu či vysoká variabilita uložených dat. Obdobně jako databáze relační i dokumentové databáze jsou primárně orientovány na data entit než na vztahy mezi nimi.

Mezi další možnosti uložení patří například *key-value* databáze či *wide column* databáze. Jejich použití je ovšem velmi specifické (např cache, uchování a zpracování velkých objemů dat) a jsou tedy vhodné pro jiné účely než je cílem této práce. Proto není těmto ostatním databázovým systémům v této kapitole věnováno více prostoru.

1.5 Analýza metod doporučování obsahu

1.5.1 Elasticsearch

Jelikož je proces hledání doporučení blízký procesu, který vykonávají vyhledávače, v rámci analýzy byl věnován čas i prozkoumání možností vyhledávacího enginu, konkrétně enginu Elasticsearch. Problém hledání souvisejících příspěvků je převoditelný na problém vyhledávání. [8] V obou případech jde o využití nějaké metriky pro ohodnocení míry shody a následné seřazení výsledků.

Elasticsearch je fulltextový vyhledávací engine. Za účelem fulltextového vyhledávání umožňuje provádět analýzu dat a provádět nad nimi nejrůznější dotazy. Data, s kterými Elasticsearch pracuje, jsou uchována v jeho interní databázi ve formátu JSON. Spolu se samotným enginem se pojí i nástroj Kibana, který poskytuje uživateli uživatelské prostředí, ve kterém může například vytvářet vizualizace z dat či data prozkoumávat.

Elasticsearch umožňuje provádět různé typy dotazů. Jedním z takových dotazů je *bool query*. Tato query umožňuje kombinovat poddotazy do jednoho. To může být využitelné právě při problému hledání doporučení, kdy dotaz není tvořen jedním řetězcem, tak jak je tomu u klasického fulltextového vyhledávání. V případě hledání doporučení je totiž dotaz tvořen množinou informací, které lze získat o uživatelových preferencích.

Nevýhodou použití enginu Elasticsearch je nízká možnost úpravy procesu hledání doporučení v případě specifických požadavků. Výhodou pro doporučování příspěvků ovšem je, že jde o do značné míry hotové řešení, které vyžaduje pouze formulaci dotazu a samotný engine se o výpočet doporučení sám postará.

1.5.2 Hledání doporučení pomocí procházení grafu

Hledání souvisejících příspěvků lze realizovat jednoduchým procházením grafu. Je možné vytvořit mnoho dotazů, jak z grafu získat tématicky příbuzné příspěvky k aktuálně zobrazenému příspěvku. Jako příklad je možné uvézt dotaz „získej všechny sousední příspěvky ve vzdálenosti 1 od aktuálně zobrazeného příspěvku“. Takový dotaz velice pravděpodobně vrátí příspěvky, které jsou opravdu tématicky blízké k aktuálně zobrazenému. Bohužel má ale takový jednoduchý dotaz značné limitace. Bere v potaz pouze příspěvky ve vzdálenosti jedna, zároveň je ale staví na stejnou úroveň bez ohledu na typ vztahu.

Pro odstranění takových limitací je možné vytvořit další podobné dotazy, které budou graf procházet jiným způsobem. Zde jsou uvedeny příklady takovýchto dotazů:

- Všichni potomci aktuálního příspěvku v hierarchii článků
- Všechny příspěvky ve stejné vzdálenosti od kořene stromu
- Všechny příspěvky z podstromu, jehož kořenem je aktuální příspěvek
- Všechny příspěvky se společnou kontaktní osobou

Problém, kdy jeden dotaz staví na stejnou úroveň všechny vrácené články, zde částečně zmizí zkombinováním takových dotazů do jednoho. Pokud by byl příspěvek X obsažen ve výsledcích takových dotazů častěji než příspěvek Y, lze uvažovat, že příspěvek X souvisí s aktuálním příspěvkem silněji než příspěvek Y.

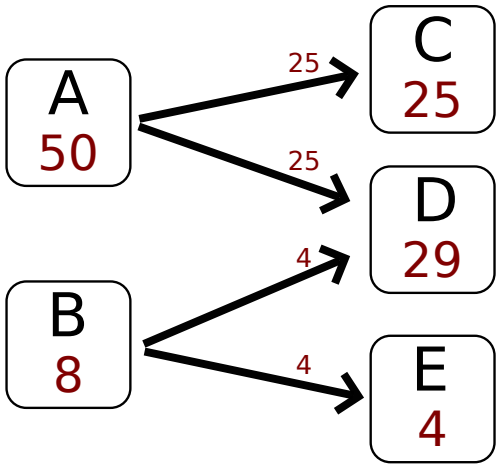
U takového řešení ale zůstává problém, že aby kombinace dotazů poskytovala dostatečně reprezentativní metriku pro seřazení seznamu doporučení, bylo by nutné sestrojit vysoké množství poddotazů a zohledňovat rozdílné typy vztahů mezi příspěvky.

1.5.3 PageRank

PageRank je grafový algoritmus vyvinutý Larry Pagem a Sergeyem Brinem v devadesátých letech minulého století. Ohodnocuje důležitost webových stránek v rámci množiny webových stránek na základě odkazů mezi nimi. [9] Výstupem algoritmu je ohodnocení webových stránek (vrcholů v grafu) určující jejich důležitost a vzájemný vliv na ostatní stránky. Kromě ohodnocení důležitosti webových stránek má PageRank využití i v oblastech jako je chemie, biologie, literatura a další. [10]

Hlavní myšlenka PageRanku tkví v tom, že množství odkazů vedoucích na danou webovou stránku pozitivně ovlivňuje její důležitost v rámci webu. [9] PageRank tedy zohledňuje strukturu grafu včetně orientace hran. Dále platí, že čím důležitější je stránka, ze které odkaz vede, tím větší má odkaz dopad na důležitost cílové stránky. Tuto myšlenku lze ilustrovat na obrázku 1.3.

Na obrázku 1.3 lze vidět webové stránky a odkazy mezi nimi znázorněné šipkami. Uvažujme, že stránky A a B již mají vypočtenou hodnotu své důležitosti. Z nich vedoucí odkazy dávají důležitost stránkám, na které vedou. Tato důležitost je pro všechny odkazy dané stránky rozdělena rovnoměrně. Jelikož ze stránky A vedou dva odkazy, každý z nich předává poloviční důležitost stránky A. Stejný případ platí pro stránku B. Důležitost stránek C, D a E je vypočtena jako součet důležitostí přicházejících odkazů. Stejným principem je důležitost předávána dále po celém grafu.



Obrázek 1.3: Princip přenášení důležitosti stránek v PageRanku

Hodnota důležitosti R stránky je zjednodušeně popsána vzorcem 1.1. Nechť u je webová stránka. B_u je množina stránek, které na stránku u odkazují. N_v je počet všech odkazů mířících ze stránky v. Zlomek $\frac{R(v)}{N_v}$ tedy říká, kolik důležitosti přenáší daný odkaz mezi stránkami. Důležitost $R(u)$ je tedy součtem důležitostí předaných přes příchozí odkazy.

$$R(u) = \sum_{v \epsilon B_u} \frac{R(v)}{N_v} \tag{1.1}$$

Limitací algoritmu PageRank je, že analyzovaný graf musí být souvislý. Pokud v grafu existuje izolovaná komponenta, PageRank se k ní nedostane, a tudíž zůstanou všechny vrcholy této komponenty ohodnoceny nulovou důležitostí. [11]

Pochopení PageRanku na příkladu náhodných návštěvníků

Význam PageRanku lze intuitivně popsat na případě takzvaných *náhodných návštěvníků* (random surfers). [9] Uvažujme, že na každou stránku vstupuje množství návštěvníků. Ti následně klikají na náhodné odkazy a pohybují se tak po webu. U každého návštěvníka existuje nenulová pravděpodobnost, že

se na stránce, kterou navštívil, zastaví. Pod tím je možné si představit situaci, že návštěvník našel, co hledal, a tím pro něj končí práce.

Na takové procházení lze nahlížet iterativně. Při každé iteraci se *náhodný návštěvník*, který ještě neukončil svou cestu, přesune pomocí odkazu náhodně na další stránku. Po několika iteracích již bude většina návštěvníků zastavena na své stránce. V závislosti na pravděpodobnosti ukončení návštěvníkovy cesty se po určitém počtu iterací ustálí počty návštěvníků na každé stránce. Počty návštěvníků na každé stránce jsou metrikou, která podle PageRanku určuje důležitost stránky.

Personalised PageRank

PageRank lze modifikovat na případ, kdy *náhodní návštěvníci* přicházejí pouze na jednu stránku z celé množiny. [9] Výsledkem nebude ohodnocení důležitosti stránek vzhledem k celému webu ale důležitost vzhledem k zdrojové stránce, do které proudí tok návštěvníků. Taková modifikace je nazývána Personalised PageRank a je využívána například sociální sítí Twitter pro hledání doporučených uživatelů ke sledování. [12]

Kapitola 2

Návrh řešení

Návrh řešení řeší otázku, jak zkonstruovat proces, na jehož začátku stojí data uložená v databázi Drupalu a na konci je samotný seznam doporučených příspěvků. Pro získání seřazeného seznamu doporučených příspěvků je nutné mít metriku, pomocí které bude možné ohodnotit míru souvislosti s aktuálně zobrazeným příspěvkem. Volba takové metriky a technologie, která podporuje její použití, je popsána v následující podkapitole.

2.1 Použité technologie a volba metriky

2.1.1 PageRank

Pro provádění analýzy souvislostí a získání doporučených příspěvků byla zvolena metrika, kterou poskytuje algoritmus PageRank. Pro tento případ se obzvláště hodí jeho modifikace Personalised PageRank. Od základní verze PageRanku se liší tím, že ohodnocuje důležitost stránek ve vztahu k pouze jedné stránce a ne ve vztahu k celému webu. Touto jednou stránkou je v tomto případě aktuálně zobrazený příspěvek. PageRank navíc umožňuje zohledňovat typy vztahů, které se v databázi fakultního webu nacházejí, na základě rozdílných vah hran v grafu. [13]

2.1.2 Neo4j

Aby bylo možné algoritmus Personalised PageRank nad daty spustit, je nutné zkonstruovat graf. Již samotná data v databázi Drupalu tvoří graf, jehož vrcholy jsou tvořeny příspěvky a hrany vztahy mezi nimi. Pro průchod tímto grafem by ale bylo nutné vykonat v databázi mnoho spojení, což může mít negativní vliv na rychlost. Proto je vhodnější použít grafovou databázi, která je pro takové operace přizpůsobena.

Konkrétní volba padla na grafovou databázi Neo4j. K výběru tohoto databázového systému přispěla jeho vysoká popularita a hodnocení na serveru

db-engines.com. [14] Ten vyvinul metodu určení popularity na základě různých ukazatelů jako například množství zmínek na webu či frekvence dotazů týkajících se daného databázového systému ve vyhledávačích. [15] Dalšími aspekty, které hrály v prospěch systému Neo4j byla rozsáhlost a podrobnost jeho dokumentace a existence pluginů, které přinášejí možnost spouštět nad grafem různé grafové algoritmy jako například zmiňovaný PageRank.

2.1.3 Modul pro systém Drupal

Funkcionalitu redakčního systému Drupal je možné rozšířit pomocí modulu, který může vykonávat libovolný kód a pracovat s vnitřními daty pomocí Drupal API. Ten bude obstarávat veškerou komunikaci s databází Neo4j.

Modul bude zároveň poskytovat možnost umístit na webové stránky takzvaný *blok*. Optikou Drupalu se v případě *bloku* jedná o HTML element, který je možné umístit kdekoliv na webovou stránku. Prostřednictvím *bloku* budou návštěvníkům prezentovány doporučení na související příspěvky. *Blok* lze volitelně konfigurovat v administraci Drupalu.

Pomocí správce závislostí Composer je dále možné do projektu přidat jakýkoliv balíček z repozitáře Packagist.org. V případě požadavků této práce se bude jednat zejména o driver zprostředkující komunikaci Drupalu s Neo4j.

2.2 Proces získávání doporučení

Je předpokládáno, že data se v databázi fakultního webu nebudou měnit příliš často a rozsáhle. Navíc není nezbytné, aby se doporučení aktualizovala při každé změně dat v databázi fakultního webu. Proto není nutné při každém dotazu na doporučení příspěvků čerpat z aktuálních dat.

Je ovšem nutné počítat s tím, že k dotazům na doporučení bude docházet často, je s tím proto potřeba v návrhu počítat a optimalizovat systém pro časté dotazování. Míru souvislosti by bylo možné vypočítat předem a pro další použití uložit doporučení do databáze. Vzhledem k rychlému provedení algoritmu PageRank, které bylo zjištěno (okolo 40 ms), ale tato optimalizace není nutná.

Časově náročnější operací je sestavení grafu příspěvků, jelikož je pro to potřeba přečíst články z databáze Drupalu a následně je včetně jejich vztahů zapsat do grafové databáze. Z důvodu optimalizace pro časté dotazování se doporučených příspěvků je proces, na jehož začátku stojí data z databáze Drupalu a na jehož konci je samotný seznam doporučení, rozdělen do dvou fází.

První fází je čtení dat z databáze Drupalu a jejich transformace do podoby, nad kterou lze efektivně aplikovat metriku pro získání doporučení. Druhou fázi tvoří dotaz Drupalu na doporučení pro aktuálně zobrazený příspěvek a jejich prezentace uživateli. Pro účely této práce vznikne modul pro Drupal, který se bude starat o komunikaci mezi Drupalem a databází Neo4j. Modul figuruje

v obou fázích procesu hledání doporučení. Tyto fáze jsou detailněji popsány v následujících dvou podkapitolách.

2.2.1 Uložení dat do grafové databáze

Aby bylo možné nad daty efektivně vykonat algoritmus PageRank, je nutné je nejdříve přetransformovat do vhodné podoby do grafové databáze. Jak již bylo řečeno, není vždy nutné vytvářet doporučení nad aktuálními daty, proto je přípustná menší odchylka mezi daty v Drupalu a daty v grafové databázi. Aby se změny v datech redakčního systému projevily v doporučeních, grafová databáze bude s daty z Drupalu v pravidelných intervalech (například 1 den) synchronizována. Opakované provádění této fáze bude zajištěno pomocí nástroje CRON, který Drupal podporuje.

O synchronizaci se bude starat modul pro Drupal vytvořený pro řešení problému, kterým se zabývá tato práce. Jelikož je v současné situaci v databázi zhruba dvě stě příspěvků, pro jednoduchost bude postačovat synchronizace způsobem, kdy nejdříve dojde k vymazání dat z grafové databáze a následně budou data z Drupalu nahrána do čisté grafové databáze. Za tento proces bude zodpovídat zmíněný modul, který projde databázi Drupalu a nalezne potřebné entity a vztahy mezi nimi užitečné pro hledání doporučení. Tato data následně zapíše do grafu v Neo4j.

Tím vznikne graf, který je možné efektivně analyzovat pomocí algoritmu PageRank implementovaného v pluginu Graph Algorithms. Ten je nutné do Neo4j doinstalovat. Vzniklý graf je výstupem první fáze procesu doporučování. Druhá fáze řeší, jak z takového grafu získat samotná doporučení.

2.2.2 Hledání doporučených příspěvků v grafu

Druhá fáze procesu získávání doporučení pokrývá proces od příchodu návštěvníka na webovou stránku po zobrazení seznamu s doporučovanými příspěvky. Hlavní roli v této fázi opět hraje vytvořený modul pro Drupal, jehož úkolem je nejdříve zjistit, na jakém příspěvku se návštěvník nachází. Tuto informaci následně modul použije pro získání doporučených příspěvků.

Aby byl spuštěn PageRank nad daty v grafové databázi, je třeba sestavit dotaz v jazyce Cypher, který spustí algoritmus s požadovanými parametry jako je počáteční vrchol (aktuálně zobrazený příspěvek) a další konfigurace potřebná pro běh algoritmu.

Modul takový dotaz předá databázi, která provede potřebné výpočty k provedení algoritmu a vzniklé výsledky vrátí seřazené sestupně dle míry souvislosti. Seznam doporučení z PageRanku bude zkombinován se seznamem příspěvků, které jsou manuálně doporučovány tvůrci webu. V momentě, kdy Drupal získá výsledná doporučení, je vykreslí pomocí šablonovacího systému Twig do HTML *bloku*. Vzniklý *blok* následně Drupal zakomponuje do webové stránky tak, jak administrátor webu nakonfiguroval v administraci Drupalu.

Celá tato fáze bude vykonávána při každé návštěvě jakéhokoliv příspěvku na webu. Protože se na vykreslení *bloku* (a tedy i získání doporučení z databáze) čeká, než je výsledný HTML kód vrácen návštěvníkovi, je nutné tuto fázi uzpůsobit pro časté čtení, aby nezpůsobovala prodlevu při načítání stránky.

Jelikož při testovacích dotazech do grafové databáze v rámci analýzy bylo dosaženo velice krátkých časů, bylo zvoleno toto řešení používající vykreslení na straně serveru. Kdyby doba potřebná k získání doporučení přesahovala mez, která by viditelně negativně působila na odezvu stránek, bylo by nutné přistoupit k řešení zahrnující například asynchronní nahrání doporučovaných příspěvků AJAX requestem.

2.3 Metrika pro doporučování

Jak již bylo řečeno v kapitole 2.1, jako metriku stanovující míru souvislosti mezi příspěvky byl zvolen algoritmus Personalised PageRank. Jeho správné použití je podmíněno existencí vhodně zvolených dat a správnou konfigurací parametrů.

2.3.1 Váhy hran

Algoritmus prochází orientovaný graf a přiřazuje jednotlivým vrcholům ohodnocení, které určuje jejich důležitost. Důležitost je přenášena z vrcholu na vrchol pomocí hran mezi nimi.

Pokud z vrcholu vede více výstupních hran (odchozích odkazů v případě webových stránek), přenášená důležitost se dělí rovnoměrně dle počtu těchto hran. Pro případy, kdy si hrany v grafu nejsou rovnocenné, zde existuje možnost udělit hranám rozdílnou váhu. [13] Čím větší váhu bude hrana mít, tím větší bude mít vliv na přenos důležitosti z vrcholu.

V případě dat, se kterými tato práce pracuje, nastává přesně takový případ, kdy si všechny hrany nejsou rovny. V databázi existuje 11 typů vztahů, kterým je možné stanovit jinou váhu pro výpočet PageRanku. Volba vah hran by se měla zakládat na důležitosti vztahů v grafu. Tímto způsobem bude možné předat PageRanku informace o významech, který mají vztahy mezi příspěvky v lidském pojetí dat. Samotná volba vah vztahů vychází z analýzy dat a je podrobněji řešena v kapitole Implementace.

2.3.2 Damping factor

Jak bylo nastíněno v kapitole 1.5.3, PageRank lze intuitivně pochopit na příkladu takzvaných *náhodných návštěvníků*, kteří cestují webem a klikají náhodně na odkazy. Dle teorie stojící za PageRankem je předpokládáno, že takový návštěvník dříve či později klikání ukončí a skončí tak na některé ze stránek. [9] Pro simulaci návštěvníků, kteří se dříve či později na některé stránce zastaví, využívá algoritmus parametr zvaný *damping factor*.

Damping factor udává, s jakou pravděpodobností návštěvník nezůstane na aktuální stránce a vydá se po některé z hran dál. [16] Obdobně lze říci, že hodnota $(1 - dampingFactor)$ říká, s jakou pravděpodobností návštěvník na aktuální stránce ukončí svou cestu. Protože se jedná o hodnotu udávající pravděpodobnost, její hodnota se pohybuje v intervalu od 0 do 1.

Přičemž v případě, kdy $dampingFactor = 0$ platí, že návštěvník ukončí cestu hned na stránce, do které vstoupil, a vztahy mezi stránkami zůstanou naprosto nezohledněny. Zdrojová stránka získá důležitost 1, důležitost všech ostatních stránek zůstane nulová.

V opačném případě, kdy $dampingFactor = 1$ dojde k tomu, že *náhodný návštěvník* bude cestovat grafem stránek do nekonečna a hodnoty důležitosti stránek nebudou konvergovat k výsledným hodnotám.

Pohybováním s hodnotou *damping factoru* se tedy upravuje chování *náhodných návštěvníků*. Vyšší hodnota *damping factoru* v případě Personalised PageRanku znamená, že důležitost článků bude rozprostřena dále od aktuálně zobrazeného příspěvku. Dle dokumentů popisujících princip algoritmu PageRank se standardně využívá hodnota 0,85. [16]

2.4 Data pro analýzu souvislostí

2.4.1 Data uložená do grafové databáze

K analýze důležitosti vrcholů pomocí algoritmu PageRank je třeba sestavit orientovaný graf. Ten se bude skládat z vrcholů tvořených příspěvky a z hran tvořených vztahů mezi nimi. Jelikož PageRank analyzuje důležitost vrcholů dle jejich pozice v grafu, důležité jsou zejména vztahy mezi příspěvky a jejich typy. Typy vztahů jsou pro PageRank užitečné pro stanovení rozdílných vah hran, které je možné do výpočtu algoritmu zapojit.

Seznam typů vztahů použitých pro analýzu algoritmem PageRank, které budou ukládány do grafové databáze:

- *primární štítek*
- *doporučený obsah*
- *místo*
- *článek* (mezi příspěvkem typu *událost* a *článek*)
- *následující*
- *předcházející*
- *průnik štítků*
- *sekundární štítky*

- *souvislosti*
- *kontakt*
- *zodpovědná osoba*

Do grafu není potřeba zapisovat data, která nejsou využitelná z pohledu analýzy souvislostí. Z tohoto důvodu může graf postrádat některá data uložená na příspěvcích, protože ta nejsou PageRankem zohledněna. Pokud by se v budoucnu ukázalo, že samotný graf uložený v grafové databázi bude analyzován kromě počítače i člověkem, je možné graf doplnit i o další data z databáze Drupalu pro jeho snadnější pochopení a orientaci v něm.

Pro identifikaci vrcholu v grafové databázi bude využit unikátní identifikátor vrcholu, který je přiřazen každému vrcholu v systému Drupal. [17] Konkrétně jde o takzvané *NID*, tedy „Node ID“.

2.4.2 Data čtená z grafové databáze

Druhá fáze procesu hledání doporučení popsaná v kapitole 2.2.2, tedy již samotné dotazování se na seznam doporučení, začíná při návštěvě příspěvku uživatelem. Při ní dojde k odeslání požadavku na server. Z tohoto požadavku Drupal získá *NID* požadovaného příspěvku. Následně dojde k dotázání se grafové databáze na doporučované příspěvky pro příspěvek s tímto *NID*.

Grafová databáze provede výpočet PageRanku pro zadaný příspěvek a vrátí seznam několika *NID* příspěvků, které PageRank ohodnotil nejlépe. Není nutné vracet kompletní seznam výsledků. Počet vrácených *NID* bude korespondovat s počtem příspěvků, které se budou zobrazovat návštěvníkovi. Počet takových příspěvků byl zvolen na pět, jelikož se jedná o číslo, které nabízí dobrý počet alternativ, kam se může návštěvník vydat. Zároveň se nejedná o příliš vysoký počet, který by návštěvníkovi trvalo přečíst.

Získaný seznam *NID* bude předán do šablonovacího systému Twig, který ze získaných *NID* vytvoří odkazy URL směřující na dané příspěvky.

Kapitola 3

Implementace

V této kapitole jsou popsány postupy a rozhodnutí spojené s implementací řešení. Samotná implementace vychází z návrhu řešení, ve kterém byly zodpovězeny zásadní otázky týkající se architektury systému a jeho vnitřního fungování.

3.1 Příprava vývojového prostředí

3.1.1 Drupal

Jak již bylo vysloveno v návrhu řešení, hlavní roli v celém procesu doporučování příspěvků bude hrát modul pro Drupal. Pro vývoj modulu pro Drupal bylo nejdříve nutné zprovoznit vývojové prostředí. Hlavním úkolem bylo zprovoznit lokální instanci Drupalu, ve které bude modul vyvíjen.

Jelikož fakultní web běží na Drupalu ve verzi 8, v této verzi byl spuštěn Drupal i v lokálním prostředí. Redakční systém Drupal je napsán v programovacím jazyce PHP a postaven na frameworku Symfony. Stažení Drupalu je možné provést pomocí správce PHP závislostí Composer příkazem:

```
$ composer create-project drupal-composer/drupal-project:8.x-dev
    fit_drupal --no-interaction
```

Tento příkaz provede stažení balíčku `drupal-composer/drupal-project` a následně provede instalaci závislostí projektu pomocí `composer install`.

Pro spuštění Drupalu ve vývojovém prostředí stačí použít PHP built-in webserver, který je součástí PHP CLI. Spustit server lze jednoduchým příkazem:

```
$ php -S localhost:8000
```

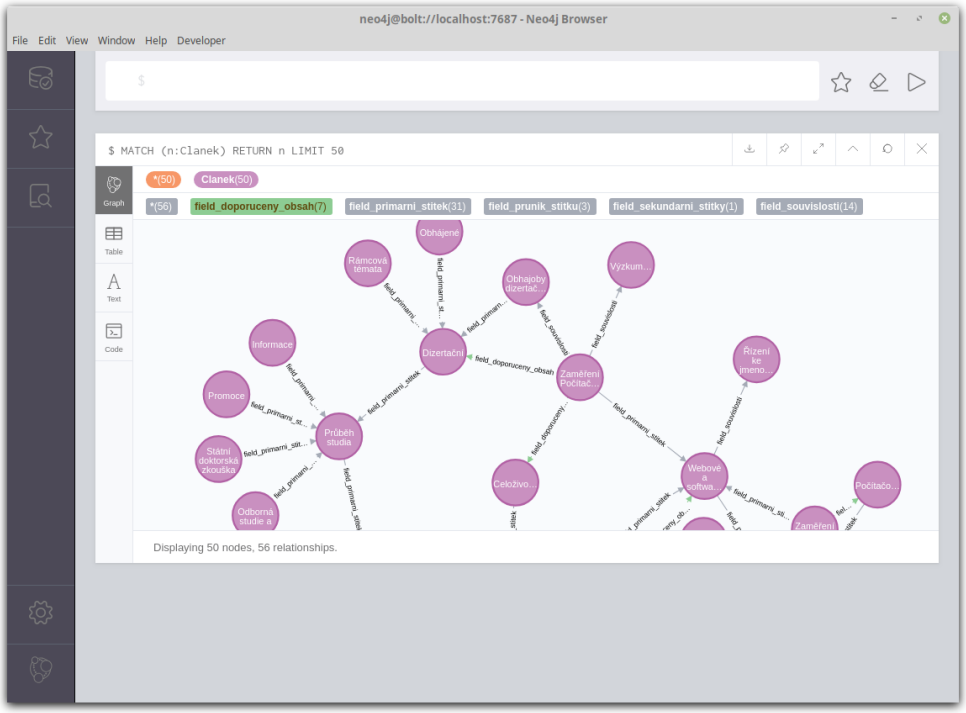
Po spuštění serveru na zvolené adrese je možné pomocí webového prohlížeče přistupovat na server.

Po otevření adresy `localhost:8000` ve webovém prohlížeči se již otevře stránka, která uživatele provede instalací samotného Drupalu. Tato část procesu zahrnuje konfiguraci instance Drupalu, připojení do databáze a vytvoření databázového schématu.

Redakční systém Drupal ke svému chodu vyžaduje databázi. Pro lokální instanci Drupalu byla zprovozněna databáze MySQL ve verzi 5.7. S databázovým schématem není nutné jakkoliv manipulovat, Drupal si jej během své instalace vytvoří sám.

3.1.2 Neo4j

Pro spuštění grafové databáze byl použit databázový engine Neo4j v edici Neo4j Enterprise for Developers. Oproti běžné Enterprise edici se vyznačuje tím, že v instalaci zahrnuje i klientskou aplikaci Neo4j Browser, pomocí které lze databázový server jednoduše konfigurovat. Tato aplikace dále umožňuje spouštět nad databází dotazy v jazyce Cypher a získané výsledky vizualizovat. Ukázka uživatelského rozhraní a vizualizace grafu je ukázána na obrázku 3.1.



Obrázek 3.1: Neo4j Browser

Pro použití algoritmu PageRank je nutné do databáze doinstalovat plugin Neo4j Graph Algorithms. Toho je jednoduché docílit opět pomocí aplikace

Neo4j Desktop. K použití pluginu není nutná žádná konfigurace, pouze je nutné restartovat databázový server.

3.1.3 Příprava dat pro vývoj

Dalším důležitým krokem bylo získání reálných dat, nad kterými by byla prováděna analýza souvislostí. Tato data mi poskytl technický správce Ing. Karel Papež v podobě MySQL dumpu popisující databázové schéma a uložená data.

Nastala ovšem komplikace, kdy nebylo možné rozběhnout lokální instanci Drupalu nad databází s daty ze získaného MySQL dumpu. Tento problém byl způsoben tím, že lokální instalace se neshodovala s instalací Drupalu připravovaného fakultního webu. Konkrétně šlo o to, že v lokální čisté instalaci Drupalu chyběla veškerá konfigurace a moduly, které byly v případě fakultního webu přítomny. Výsledkem bylo, že při pokusu rozběhnout lokální Drupal nad databází fakultního webu, server opakovaně vracel error `HTTP 500`.

Aby se data z dumpu dostala do lokální instance Drupalu, zvolil jsem cestu načíst z MySQL dumpu fakultního webu pouze data týkající se příspěvků a následně je zapsat do databáze lokální instance Drupalu pomocí Drupal Entity API. Pro tento účel byl vytvořen skript, který projde strukturu uložených dat ve zdrojové databázi a na výstupu poskytne tato data ve formátu JSON.

Zde je důležité zmínit, že struktura dat uložených v databázi Drupalu není v podobě, kdy jeden řádek tabulky odpovídá jedné entitě. Jak již bylo řečeno v kapitole 1.4.1, Drupal nahlíží na všechny uložené objekty jako na vrcholy (nodes). Drupal vytváří pro každý typ pole svou vlastní tabulku. V řádce takové tabulky je kromě ostatních informací uložena samotná hodnota a odkaz na příslušný vrchol ve formě *NID*. Pomocný program použitý pro čtení dat z MySQL databáze musel s touto skutečností pracovat.

Serializovaná data v JSON formátu bylo následně nutné zapsat do lokální instance Drupalu. Toho bylo docíleno vytvořením pomocného modulu, pomocí nějž byla přečtena data ve formátu JSON a následně byla pomocí Drupal Entity API zapsána do databáze čisté instalace Drupalu. Procedura importu byla vytvořena tak, aby ji bylo možné spustit z příkazové řádky pomocí příkazu `$ drush fitjsondata:import`. Na konci tohoto procesu tedy stála lokální instalace Drupalu s naimportovanými daty z fakultního webu.

3.2 Tvorba modulu pro Drupal

Modul Drupalu je soubor zdrojových souborů obvykle zaarchivovaný ve formátu `modul.tar.gz`. Prvním krokem k vytvoření Drupalu je zvolení jeho názvu. K názvu musí být přidružen takzvaný *machine name*, což je textový řetězec, pomocí kterého se bude Drupal vnitřně na modul odkazovat. Pro modul vytvořený pro účely této práce byl zvolen název „FIT recommendation block“, čemuž odpovídá *machine name* `fit_recommendation_block`.

Veškeré soubory modulu se po jeho instalaci do Drupalu nacházejí v adresáři `modules/fit-recommendation-block/` v kořenovém adresáři projektu. Důležitý je soubor `fit_recommendation_block.info.yml`, který v sobě nese základní informace o modulu jako například jeho jméno či verzi. Aby bylo možné přidat závislost modulu na driver *GraphAware Neo4j PHP Client*, je nutné přidat soubor `composer.json` a pomocí příkazu `$ composer require "graphaware /neo4j-php-client:^4.0"` jej do závislostí přidat.

Pro vytvoření *bloku* je součástí modulu třída `FITRecommendationBlock` v jmeném prostoru `Drupal\fit_recommendation_block\Plugin\Block`. Součástí této třídy je metoda `build`, která definuje chování *bloku*, když je uživateli zobrazen na webové stránce. V této metodě je iniciován proces hledání doporučení v databázi Neo4j

Blok je možné v administraci Drupalu konfigurovat. Pro případ, že uživatel konfiguraci nevyplní, *blok* obsahuje i výchozí konfiguraci, která bude v takovém případě využita. Výchozí konfigurace je uložena v souboru `config/install/ fit_recommendation_block.settings.yml`. Formulář pro změnu konfigurace je definován v metodě `blockForm` třídy `FITRecommendationBlock`. Tento formulář se uživateli zobrazí v administraci při úpravách *bloku*.

3.2.1 Implementace zápisu dat do Neo4j

Pokud mluvíme o zápisu dat do Neo4j, jedná se o první fázi procesu hledání doporučení. Při ní je nutné přečíst všechny potřebné entity v databázi Drupalu a zapsat je do grafové databáze. Proceduru je potřeba navázat na tzv. Drupal CRON, který je blíže popsán v podkapitole 1.4.1.

Čtení z databáze Drupalu je realizováno pomocí Drupal Entity API. Pro každý typ obsahu jsou z databáze načteny všechny publikované vrcholy. Ty jsou následně zapsány pomocí Cypher dotazu. Dotaz, který vytvoří v Neo4j vrchol typu článek s *NID* 330 vypadá takto:

```
MERGE (a:Clanek {nid: 330})
```

Po úspěšném uložení všech vrcholů je možné vytvořit v grafové databázi vztahy mezi vrcholy. Pro každý vrchol jsou do grafové databáze zapisovány všechny z něj vycházející hrany tímto způsobem:

```
MATCH (a {nid: 330}),(b {nid: 420}) MERGE (a)-[r:primarni_stitek]->(b)
```

Tato query nejdříve ve své `MATCH` části nalezne zdrojový a cílový vrchol. Pokud jsou vrcholy *a* a *b* se zadanými *NID* nalezeny, je mezi nimi pomocí `MERGE` klauzule vytvořen vztah typu `primarni_stitek`. Základní principy jazyka Cypher jsou poprány v podkapitole 3.3.1.

GraphAware Neo4j PHP Client umožňuje pro vylepšení výkonu uložení dotazů do *zásobníku* a jeho následné zpracování grafovou databází najednou

v pořadí, ve kterém byly dotazy přidány. S použitím takového *zásobníku* lze vytvořit více vrcholů pomocí následujícího PHP kódu:

```
$stack = $this->client->stack();
$query = "MERGE (a:" . $nodeType . " {nid: {nid}})";
foreach ($nodes as $node){
    $stack->push($query, ["nid" => $node->id()]);
}
$this->client->runStack($stack);
```

Celá tato procedura, která čte data z databáze Drupalu a následně je zapisuje do Neo4j, je součástí takzvaného CRON hooku. Pomocí nástroje CRON jsou v Drupalu spouštěny pravidelně se opakující úkoly jako například kontrola aktualizací. CRON hook je v Drupalu možnost, jak spustit vlastní kód při vykonání CRON procedury. CRON hook je možné realizovat pomocí vytvoření speciální funkce v souboru `fit_recommendation_block.module`. Název funkce musí být ve formátu `module machine name_cron`, v tomto případě se tedy funkce jmenuje `fit_recommendation_block_cron`. V jejím těle je kód, který je při každém spuštění *CRON* procedury spuštěn.

3.2.2 Implementace získání doporučení z Neo4j

Proces získání doporučení začíná přístupem návštěvníka na příspěvek fakultního webu. Nejdříve je potřeba identifikovat, na kterém příspěvku se návštěvník nachází. Toho je dosaženo získáním *NID* z parametrů requestu. Drupal umožňuje jednoduše číst takové parametry tímto způsobem:

```
$parameters = \Drupal::routeMatch()->getParameters();
$currentNode = $parameters->get("node");
```

Následně je s pomocí *NID* aktuálně zobrazeného příspěvku sestaven Cypher dotaz pro získání doporučených příspěvků. Sestavení tohoto dotazu, který spustí nad daty algoritmus PageRank, je popsáno dále v podkapitole 3.3. Za účelem komunikace s Neo4j a získáním výsledků z grafu je v modulu definována třída GraphReader.

V návrhu řešení bylo zvoleno, že *blok* s doporučeními bude uživateli zobrazovat pět odkazů na doporučované příspěvky. Z principu fungování Personalised PageRanku vyplývá, že zdrojový vrchol, pro nějž jsou vypočítávány důležitosti vrcholů, bývá ve výsledném ohodnocení velice často umístěn na prvních pozicích dle hodnoty důležitosti.

Z tohoto důvodu je z grafové databáze pokaždé získáván seznam doporučených příspěvků o jeden příspěvek delší, než bude zobrazeno uživateli. Děje se tak pro případ, že počáteční vrchol (aktuálně zobrazený příspěvek) bude právě obsažen v seznamu získaném z Neo4j a bude jej tedy potřeba před vykreslením uživateli vyřadit. Pokud v získaném seznamu aktuálně zobrazený

příspěvek nefiguruje, seznam je oříznut na požadovaný počet příspěvků odstraněním nejhůře ohodnoceným doporučením.

Vykreslení bloku se seznamem doporučení

Po oříznutí seznamu doporučení na počet pěti příspěvků je tento seřazený seznam *NID* předán prezentační vrstvě. O vykreslení dat do HTML se stará šablonovací systém Twig. Pro vykreslení je nejdříve potřeba šablony. Konkrétní šablona pro *FIT recommendation block* je uložena v souboru `templates/fit_recommendation_block.html.twig`. Jedná se o zdrojový kód HTML obohacený o značky Twigu. Ty jsou před odesláním výsledného HTML nahrazeny hodnotami vyhodnocenými Twigem. Tím vznikne čistý HTML kód, který je v HTTP odpovědi vrácen uživateli.

Do šablony je pomocí Twigu možné předat proměnné pomocí PHP. Hodnoty proměnných pak můžou být pomocí Twig značek dále zpracovány a vytisknuty to výsledného HTML. Kromě jednoduchého vytisknutí umožňuje Twig volat nadefinované funkce. V tomto modulu je takto z Twigu volána funkce `path`, která pro zadaný příspěvek vytvoří URL směřující na stránku s tímto příspěvek. Tímto způsobem je možné pomocí Twig šablony vytvořit HTML, které vykreslí pro každý příspěvek jiný seznam doporučení. Výsledný *blok* je zobrazen na obrázku 3.2.

Obrázek 3.2: Blok se seznamem doporučených příspěvků

3.3 Dotaz pro získání doporučených příspěvků

Pro získání seznamu doporučených příspěvků z grafové databáze je potřeba vytvořit dotaz, pro nějž provede grafová databáze potřebné výpočty. Aby bylo možné dotaz pochopit, je nutné se nejdříve jemně seznámit s dotazovacím jazykem Cypher.

3.3.1 Jazyk Cypher

Neo4j využívá dotazovací jazyk Cypher. Nejjednodušším příkladem dotazu, který z databáze přečte data, je `MATCH` klauzule. Pomocí ní je možné definovat vzor, který se bude databáze snažit najít. Vzorem můžou být například dva vrcholy spojené hranou. Vzor je v jazyce Cypher možné popsat syntaxí, která je inspirovaná ASCII artem. To znamená, že textový řetězec popisující vzor připomíná vizuálně samotný graf. Například vrchol je možné popsat kulatými závorkami takto: `(vrchol1)`, dva vrcholy propojené vztahem pak takto: `(vrchol1)--(vrchol2)`.

Pro odlišení různých typů vrcholů ale i hran lze přiřadit vrcholu či hraně svůj název (*label*). Ten lze v Cypher dotazu zohlednit při definování vzoru za dvojtečkou (např: `(clenRodiny:Osoba)`).

Pro všechny výskyty daného vzoru je následně provedeno pokračování dotazu. Příkaz bývá typicky zakončen klauzulí `RETURN`, která stanovuje, jaká data má dotaz vracet. Pro příklad uvádím jednoduchý příklad dotazu:

```
MATCH (spisovatel:Osoba)-[:je_autorem]-(dilo:Kniha) RETURN dilo,
    spisovatel.
```

Tento dotaz nejdříve projde uložená data a pokusí se v nich najít takové části grafu, které odpovídají definovanému vzoru. Každý výskyt vzoru se následně dostává do části dotazu s `RETURN` klauzulí. Výsledkem dotazu jsou dvojice tvořené všemi knihami v grafu a jejich příslušnými autory.

Filtrování vybraných výsledků na základě nějaké podmínky či řazení lze v jazyce Cypher realizovat pomocí klauzulí `WHERE` a `ORDER BY`. Pro omezení počtu výsledků lze použít klauzuli `LIMIT`. Jelikož je jejich princip podobný jako v jazyce SQL a jejich důležitost v kontextu práce není vysoká, není jim v této kapitole věnováno více prostoru.

3.3.2 Spuštění PageRanku pomocí Cypher dotazu

Standardní `MATCH` dotaz není pro realizaci algoritmu PageRank dostačující. Ke spuštění algoritmu, který je součástí doinstalovaného pluginu Graph Algorithms je nutné využít klauzule `CALL` takovýmto způsobem:

```
CALL algo.pageRank.stream('Clanek', 'LINKS', {iterations:20,
    dampingFactor:0.85})
```

Tímto způsobem je vykonána klasická verze algoritmu PageRank. Jak je z dotazu vidět, používá hodnotu 0,85 pro *damping factor* a počet iterací výpočtu PageRanku je roven 20.

Pro účely této práce je ale nutné využít modifikaci Personalised PageRank. Tato modifikace se vyznačuje tím, že je nejdříve potřeba nalézt počáteční vrchol, pro který bude PageRank stanovovat důležitost dalších vrcholů. Tento

počáteční vrchol lze nalézt pomocí jednoduché `MATCH` klauzule před spuštěním samotného PageRanku. Nalezený počáteční vrchol je následně algoritmu předán pomocí parametru `sourceNodes`.

```
MATCH (pocatecniClanek:Clanek {name: Č"lánek X"})
CALL algo.pageRank.stream('Clanek', 'LINKS', {iterations:20,
    dampingFactor:0.85, sourceNodes: [pocatecniClanek]})
```

Pro zvýšení relevance výsledků algoritmu je potřeba v dotazu zohlednit i různé typy vztahů. Toho lze docílit použitím vah hran. Jelikož i hrany mohou v grafové databázi v sobě nést data, lze do nich jednoduše zapsat samotné váhy a při spuštění PageRanku tyto váhy číst.

Existuje zde ale i možnost přiřazovat váhy vztahům až v momentě, kdy dochází k výpočtu PageRanku. Takový přístup přináší možnost měnit váhy při každém průchodu PageRanku bez změny zapsaných dat v grafu. Zapsáním vah do grafu by navíc docházelo ke zbytečné redundanci dat, jelikož jeden typ vztahu bude mít při výpočtu vždy stejnou váhu.

3.3.3 Transformace grafu pro PageRank

Pro analýzu konkrétního grafu příspěvků, se kterým tato práce pracuje, je zkreslující, že PageRank prochází graf pouze po směru orientovaných hran. Ačkoliv v databázi Drupalu je směr vztahů vždy jasně definován, pro PageRank by mělo být možné procházet hrany v obou směrech. Pro takové účely lze graf přetransformovat do podoby, kdy z jedné orientované hrany se stanou dvě orientované hrany vedoucí v obou směrech. Těmto nově vzniknuvším hranám lze opět udělit váhu.

Použití dvou opačně orientovaných hran mezi dvěma příspěvky lze dobře popsat na příkladě. Řekněme, že máme dva příspěvky, mezi kterými vede vztah typu *primární štítek*. Tento vztah bude přetransformován do dvou vztahů, kde každému z nich bude přidělena váha. Pokud bude pro vztah ve směru od rodiče k potomkovi přidělena váha hodnoty 0,7 a v opačném směru bude zvolena váha 0,3, bude tím předána PageRanku informace, aby upřednostňoval články umístěné v hierarchii níže.

Příklad vlivu vah hran na výsledky PageRanku

Na následujícím příkladě je názorně demonstrován vliv vah hran. PageRank byl aplikován na článek „Bakalářský studijní program“ („BSP“). K výpočtu byly pro jednoduchost zohledněny pouze vztahy typu *primární štítek*. V prvním případě byly váhy nastaveny v obou směrech vztahu na shodnou hodnotu 0,5. V druhém případě byly váhy upraveny na hodnotu 0,7 ve směru od rodiče k potomkovi a v opačném směru na hodnotu 0,3.

Tím pro PageRank vznikla tendence zvyšovat důležitost příspěvkům vyskytujícím se v hierarchii níže. To lze vidět na rozdílném pořadí příspěvků.

Z výsledků je zajímavé si povšimnout, že v případě dotazu s nesymetrickými hodnotami v tabulce článek „Studijní programy“, který je rodičem článku „Bakalářský studijní program“, ve výsledcích klesl níže.

1.	Bakalářský studijní program
2.	Přijímací řízení
3.	Bakalářské obory
4.	**Studijní programy** (rodič „BSP“)
5.	Bakalářská závěrečná zkouška
6.	Bakalářská práce
7.	O bakalářském programu
8.	Kombinované studium

Tabulka 3.1: Seřazený seznam s použitím symetrických vah v obou směrech

1.	Bakalářský studijní program
2.	Přijímací řízení
3.	Bakalářské obory
4.	Bakalářská závěrečná zkouška
5.	Bakalářská práce
6.	O bakalářském programu
7.	Kombinované studium
8.	**Studijní programy** (rodič „BSP“)

Tabulka 3.2: Seřazený seznam s použitím nesymetrických vah v obou směrech

Cypher projection

Takovou transformaci grafu, kdy z hrany v jednom směru vzniknou hrany dvě v opačných směrech, je možné realizovat pomocí takzvané *Cypher projection*. Jedná se o transformaci fyzicky uloženého grafu do grafu virtuálního. [18] S tímto virtuálním grafem je následně možné provádět další operace jako například spuštění PageRanku.

Virtuální graf lze sestrojit na základě existujícího grafu s použitím transformačních pravidel. Například je možné říci, že pro každý vztah typu *primární štítek* v původním grafu bude ve virtuálním grafu existovat vztah stejné orientace s váhou 0,7. Tím je vyřešen problém redundance vah uložených v grafu zmiňovaný v předchozí podkapitole. Pro umožnění PageRanku procházet hrany oběma směry je dále užitečné využít pravidlo, které by pro každý vztah v daném směru vytvořilo ve virtuálním grafu dva vztahy ve vzájemně opačných směrech.

Taková transformační pravidla použitá v *Cypher projection* lze popsat relativně jednoduchými Cypher poddotazy. Zde je příklad transformačního pravidla, které přetransformuje dva vrcholy spojené vztahem typu *primární štítek* do dvou vrcholů spojených hranou o váze 0,7. Zároveň je dobré si povšimnout, že z vrcholu `m` se stal zdroj (`id(m)AS source`) a z vrcholu `n` se stal cíl (`id(n)AS target`). To má za následek, že ve virtuálním grafu bude mít hrana opačnou orientaci než v grafu původním.

```
MATCH (n)-[:field_primarni_stitek]->(m) RETURN id(m) AS source, id(n)
    AS target, 0.7 AS weight
```

Pro účely této práce vzniklo během implementace 14 takových transformačních pravidel. Takovýto počet pravidel vznikl, protože s pomocí PageRanku bude zohledněno 9 typů vztahů. U pěti nejdůležitějších typů je ale navíc používána odlišná váha pro každý směr hrany, kvůli tomu narostl počet pravidel na 14.

Sérii takovýchto pravidel lze spojit do jednoho pravidla použitím klauzule UNION následovně: `pravidlo1 UNION pravidlo2 UNION ...`. Takto vzniklé spojené pravidlo lze předat jako argument do algoritmu PageRank. Při volání dotazu zbývá už jen sdělit PageRanku, aby pracoval s vzniklým virtuálním grafem. Výsledný dotaz má tuto podobu:

```
MATCH (subject:clanek {nid:${nid}}) CALL algo.pageRank.stream(
  'MATCH (n) RETURN id(n) as id',
  'MATCH (n)<-[:field_primarni_stitek]-(m) RETURN id(n) AS source, id(
     m) AS target, 0.38 AS weight
   UNION MATCH (n)-[:field_primarni_stitek]->(m) RETURN id(n) AS
      source, id(m) AS target, 0.48 AS weight
   UNION MATCH (n)-[:field_doporuceny_obsah]->(m) RETURN id(n) AS
      source, id(m) AS target, 0.48 AS weight

  ...

   UNION MATCH (n)-[:field_souvislosti]->(m) RETURN id(n) AS source,
      id(m) AS target, 0.16 AS weight',
  {graph: 'cypher', iterations:20, dampingFactor:0.85, weightProperty
     :'weight', sourceNodes: [subject]}
)

YIELD nodeId, score
WHERE score > 0 AND algo.getNodeById(nodeId):" . $node->bundle() . " "
    . "
RETURN algo.getNodeById(nodeId) as res, score
ORDER BY score DESC
LIMIT ${limit}
```

V takovéto podobě je nový virtuální graf vytvořen při každém zavolání dotazu. V současnosti takové řešení plně postačuje požadavkům na rychlost.

Pro případné zlepšení výkonu je ovšem možné virtuální graf načíst předem do paměti pod libovolně zvoleným jménem a pod tímto jménem se na něj v dotazu následně odkazovat.

Kapitola 4

Testování a ladění výsledků

Testování je nedílnou součástí vývoje softwarového projektu. Jeho cílem je ověřit správnou funkčnost projektu jak z pohledu správných výstupů algoritmu, tak z pohledu přínosu pro uživatele. V této kapitole je zároveň věnován prostor ladění vah pro algoritmus PageRank tak, aby poskytoval co nejkvalitnější doporučení.

4.1 Testování PHP kódu

Standardem pro testování PHP kódu je framework PHPUnit. Poskytuje programátorovi prostředí, ve kterém může jednoduše ověřovat tvrzení a kontrolovat tak výstupy kódu. Pro případy, kdy kód pracuje s objekty, jejichž chování je v testovacím prostředí složité replikovat, PHPUnit umožňuje vytvořit pro objekt takzvaný *mock*, který se navenek může chovat jako požadovaný objekt bez nutnosti řádného vnitřního chování.

Základním typem testů jsou jednotkové testy, které ověřují správnou funkčnost menších částí kódu. Jednotkové testy použité v modulu *FIT recommendation block* jsou umístěny v adresáři `tests/src/Unit`. Jejich spuštění ověřuje zejména korektní transformaci dat při sestavování seznamu doporučených příspěvků.

Druhým typem testů, které jsou v modulu použity, jsou integrační testy. Jejich cílem je zejména ověřit fungující spolupráci systému Drupal s databází Neo4j. Pro spuštění integračních testů je nutné, aby byla spuštěna Neo4j s testovacím grafem pro ověření korektní komunikace s databází.

Pro jednoduché použití frameworku PHPUnit byl vytvořen soubor `phpunit.xml.dist`, který obsahuje základní konfiguraci pro spuštění testů modulu. Podrobnější informace ohledně spuštění testů je popsáno v souboru `README.md`. Provést testy je možné pomocí následujícího příkazu:

```
$ phpunit -c fit-recommendation-block/phpunit.xml.dist
```

4.2 Uživatelské testování a ladění vah PageRanku

Cílem uživatelského testování je stanovit, zda implementované řešení poskytuje kvalitní výsledky. Především kvalitní výsledky totiž určují, zda bude mít práce pro uživatele opravdu kýžený přínos.

4.2.1 Hledání ideálních vah

Jak bylo řečeno v kapitole implementace, jedněmi z parametrů pro algoritmus PageRank jsou váhy hran, nad kterými je algoritmus proveden. Stanovit ideální hodnoty vah tak, aby PageRank poskytoval co nejlepší výsledky je ovšem složitý úkol. Mnohem jednodušší je pokusit se stanovit tyto váhy pomocí intuice na základě myšlenek formulovaných v analýze a pochopení významu vah.

Pro intuitivní stanovení vah byly dále uplatněny informace získané z konzultace s Marií Böhmovou, která se podílí na tvorbě a správě obsahu fakultního webu. Vzniklé váhy byly zvoleny takto:

`field_primarni_stitekLR`	0,35
`field_primarni_stitekRL`	0,6
`field_doporuceny_obsahLR`	0,4
`field_doporuceny_obsahRL`	0,1
`field_clanek`	1
`field_misto`	0,15
`field_kontakt`	0,35
`field_zodpovedna_osoba`	0,25
`field_prunik_stitkuLR`	0,1
`field_prunik_stitkuRL`	0,25
`field_sekundarni_stitkyLR`	0,25
`field_sekundarni_stitkyRL`	0,5
`field_souvislostiLR`	0,15
`field_souvislostiRL`	0,1

Tabulka 4.1: Intuitivně volené váhy hran pro PageRank

Některým typům vztahů je přidělena dvojice vah pro každý směr vztahu. To je v tabulce znázorněno postfixem „LR“ (*left – right*, po směru vztahu) a „RL“ (*right – left*, proti směru vztahu). Ostatní vztahy mají stanovenou shodnou váhu pro oba směry.

Intuitivně stanovené váhy hran lze sice jednoduše stanovit, ovšem od ideálních vah se mohou značně lišit. Způsob, jakým získat ideální váhy, může být projít všechny možné kombinace vah a nějakým způsobem ohodnotit kvalitu získaných výsledků. Jelikož se jedná o velký stavový prostor, je nutné jej procházet strojově.

Pro nalezení ideálních vah bylo nejdříve potřeba stanovit, jaké jsou ideální výsledky algoritmu. Za tímto účelem byly na základě konzultace s Marií Böhmovou stanoveny ideální doporučení pro pět článků z různých odvětví webu. Konkrétně vzniklo pět seřazených seznamů po deseti doporučených článcích. Cílem PageRanku tedy bylo se těmto ideálním seznamům svými výsledky co nejvíce přiblížit.

4.2.2 Metrika pro porovnání seznamu doporučení

Dále bylo potřeba vytvořit metriku, která ohodnotí, jak moc se získané výsledky podobají těm ideálním. Na vstupu do metriky stojí vždy seznam ideálních výsledků o deseti doporučeních. S nimi je potřeba porovnat seznam článků, které byly získány s pomocí algoritmu PageRank s použitím daných vah.

Pro porovnání těchto seznamů byla zvolena metrika, která se pro každé doporučení pokusí nalézt příslušné doporučení i v reálném seznamu získaném z PageRanku. Pro každý takto nalezený článek je vypočítána vzdálenost mezi pozicí v ideálním seznamu a ve reálném seznamu. Vzdálenosti pro každý článek jsou sečteny a výsledkem je skóre, které určuje míru podobnosti seznamů. U takto vypočteného skóre platí, že čím menší hodnoty nabývá, tím větší je podobnost ideálního a skutečného seznamu doporučení.

Pokud se prvních deset doporučení ve skutečném seznamu rovná ideálnímu seznamu, vzdálenost pro každý článek je nulová, a tedy i výsledné skóre je rovno nule. Naopak v nejhorším případě, kdy není PageRankem žádný z článků nalezen, je do výsledného skóre přičtena pro každý článek penalizace, která se rovná počtu článků v databázi Drupalu. Toto číslo se v čase mírně měnilo, v době psaní práce se v databázi Drupalu vyskytovalo 203 článků. V nejhorším případě tedy může skóre nabýt hodnoty $10 * 203 = 2030$.

Jelikož z konzultace ideálních doporučení vzešlo takovýchto seznamů pět, pro každé nastavení vah došlo k pěti výpočtům skóre podobnosti. Těchto pět skóre bylo vždy algoritmem sečteno. Nejhorší možnou hodnotou skóre tedy ve výsledku je $2030 * 5 = 10150$

Tato metrika byla aplikována na různé způsoby hledání doporučení. Vznikly tak celkem 4 případy, pro které bylo vypočteno výsledné skóre:

Způsob hledání doporučení	skóre
Náhodné pořadí příspěvků (průměr z 500 pokusů)	4862
PageRank s náhodnými váhami (průměr z 500 pokusů)	697
PageRank s intuitivně stanovenými váhami	562
PageRank se stejnými váhami	545

Tabulka 4.2: Skóre různých způsobů pro hledání doporučení

4.2.3 Procházení stavového prostoru vah

V návrhu řešení byla stanovena potřeba 14 vah, kde každá z nich může nabývat nějakého reálného čísla v intervalu od 0 do 1. Při n možných hodnotách pro každou váhu vzniká stavový prostor o n^{14} možných stavech. Takto vzniklý stavový prostor je rozsáhlý a vyžadoval by větší množství času pro otestování jeho možných kombinací hodnot.

Proto bylo zvoleno sestavit heuristiku, která vypočte uspokojivé nastavení vah při nízkém počtu otestovaných stavů. Byly sestaveny tři vzájemně podobné heuristiky.

Procházení stavů tvořených náhodně generovanými vahami

První heuristikou bylo generovat váhy náhodně a pro každý náhodně vygenerovaný stav vypočítat výsledné skóre podobnosti seznamů doporučení. Z takto provedených výpočtů by následně byl vybráno nastavení vah, na základě kterých vznikla doporučení s nejnižším skóre. Pro 500 výpočtů skóre bylo vypočteno nejnižší skóre 394.

Hledání vah postupným procházením hodnot

Na počátku druhé heuristiky byly váhy stanoveny na počáteční hodnotu 0,5. Heuristika byla navržena tak, aby postupně prošla všech 14 vah a pro každou váhu otestovala několik hodnot jdoucích po sobě po stejných skocích (například 0,1, 0,2, 0,3, …, 1). Pro každou váhu byla následně uložena hodnota, která poskytnula co nejlepší výsledné skóre. Nad kompletním seznamem takto vzniklých nejlepších hodnot byl nakonec opět proveden PageRank a vypočteno jeho skóre.

Takto vzniklé váhy je možné použít jako počáteční váhy a výpočet heuristiky s nimi opakovat. Po opakovaném provedení této heuristiky ovšem není zaručeno dosažení lepšího výsledku. Doporučení na základě takto vzniklých vah měla po třech opakováních metriky hodnotu skóre 415. Po čtvrtém opakování výpočtu metriky se ale skóre navýšilo na 525.

Opakování heuristiky	skóre
První opakování	462
Druhé opakování	434
Třetí opakování	**415**
Čtvrté opakování	525

Tabulka 4.3: Vývoj skóre při opakování výpočtů heuristiky bez zapojení získaných vah do výpočtu

Hledání vah postupným procházením hodnot s jejich okamžitým zapojením do výpočtu

Třetí heuristika je principiálně velmi podobná předchozí heuristice. Rovněž prochází seznam 14 vah postupně. Na rozdíl od předchozí heuristiky si ale nejlepší váhy neukládá stranou, ale po jejich získání je okamžitě zapojí do výpočtu. Díky tomu se hodnoty vah mění tak, aby bylo vždy dosaženo lepšího nebo stejného skóre. Tím se liší od předchozích heuristik, u kterých platí, že skóre použitých vah se může v průběhu výpočtu zhoršit. Po pěti opakováních heuristiky se skóre ustálilo na hodnotě 330.

Opakování heuristiky	skóre
První opakování	406
Druhé opakování	358
Třetí opakování	345
Čtvrté opakování	**330**
Páté opakování	330

Tabulka 4.4: Vývoj skóre při opakování výpočtů heuristiky se zapojením získaných vah do výpočtu

4.2.4 Volba výsledných vah

Během výpočtů heuristik se často ukazovalo, že některé váhy hrají na výsledném skóre (respektive na výsledných doporučeních) nízkou roli. Skóre totiž zůstávalo na téměř stejných hodnotách bez ohledu na měnící se hodnotu těchto vah. Z tohoto důvodu se často stávalo, že tyto váhy zůstaly na hodnotách velice blízkých nule, protože algoritmus provádějící heuristiku s jejich měnícími se hodnotami nepozoroval vylepšení výsledného skóre. Jednalo se zejména o váhy použité ve vztazích:

- *článek* - *místo*,
- *článek* - *událost*,
- *článek* - *kontakt*.

Nízký dopad vztahu *článek* - *místo* na výsledky PageRanku si vysvětluji tím, že v databázi Drupalu existuje velice malé množství entit typu *místo*. Pro zvýšení jeho významu by bylo potřeba doplnit větší množství dat a vztahů týkající se míst.

V porovnání s entitou typu *místo* je počet událostí a osob v grafu mnohem vyšší. Důvodem nízkého dopadu vztahu *článek* - *událost* a *článek* - *kontakt* může být to, že *osoba* a *událost* působí v grafu velice často jako jeho listy.

Kvůli tomu se jejich získaná důležitost nemá kam dále šířit a ovlivňovat tak ohodnocení PageRanku dalších vrcholů v grafu. Na následující tabulce lze pozorovat, že oproti článkům lze u osob a událostí opravdu pozorovat větší procento vrcholů, které jsou listy grafu.

Typ vrcholu	Celkový počet	Počet listů	Procento listů
osoba	256	176	68.75
událost	88	30	37.5
místo	9	2	22.22
článek	203	4	1.97

Tabulka 4.5: Porovnání typů obsahu z hlediska množství listů v grafu

Pro výsledné použití v modulu *FIT recommendation block* byly použity váhy, které vzešly z výpočtu třetí metriky, jelikož získaly nejlepší skóre. Hodnoty, které měly na výsledky PageRanku nízký dopad, a algoritmus jim proto přiřkl nízké hodnoty, byly upraveny na základě intuitivně vzniknuvších vah. To bylo učiněno pro případ, že budou data v databázi Drupalu v budoucnu obohacena tak, že i tyto vztahy získají pro PageRank větší význam.

Pro výsledné váhy byly nakonec zvoleny tyto hodnoty:

`field_primarni_stitekLR`	0,48
`field_primarni_stitekRL`	0,38
`field_doporuceny_obsahLR`	0,48
`field_doporuceny_obsahRL`	0,88
`field_clanek`	0,48
`field_misto`	0,14
`field_kontakt`	0,2
`field_zodpovedna_osoba`	0,52
`field_prunik_stitkuLR`	0,34
`field_prunik_stitkuRL`	0,4
`field_sekundarni_stitkyLR`	0,15
`field_sekundarni_stitkyRL`	0,78
`field_souvislostiLR`	0,16
`field_souvislostiRL`	0,1

Tabulka 4.6: Intuitivně volené váhy hran pro PageRank

4.2.5 Testování vylepšených vah

Vzniklé váhy bylo potřeba otestovat, zda opravdu poskytují lepší výsledky než váhy, které byly stanoveny intuitivně. Za tímto účelem vznikly dvě sé-

rie seznamů doporučených příspěvků. Pro pět článků z různých oblastí webu byly vygenerovány seznamy doporučení na základě intuitivních vah a na základě vylepšených vah, které vzešly z procházení stavového prostoru pomocí vytvořené heuristiky.

Takto vzniklé seznamy byly postaveny vedle sebe a konzultovány s Marií Böhmovou, který ze seznamů poskytuje lepší doporučení. Zároveň nebylo řečeno, který ze seznamů představuje intuitivně vzniklé váhy a který váhy vylepšené. V 60% případů byl seznam získaný s pomocí nově vzniklých vah označen za lepší ze dvou porovnávaných.

Na základě tohoto čísla nelze s jistotou říci, zda nově vzniklé váhy poskytují lepší výsledky oproti výsledkům vzniklých na základě intuitivně stanovených vah. Samotné skóre určující podobnost se seznamem ideálních doporučení se ovšem viditelně zlepšilo (562 s intuitivními váhami, 330 s nově vzniklými váhami).

Hlavní zlepšení kvality doporučení ale přineslo hlavně samotné použití algoritmu PageRank (zřetelné v tabulce 4.2). Hledáním ideálních vah posléze nedošlo k výraznému zlepšení kvality doporučení ale pouze k mírnému posunu, což dokumentuje zlepšení skóre podobnosti s ideálními doporučeními o 232 bodů.

Závěr

V rámci této práce se podařilo zanalyzovat strukturu obsahu fakultního webu a naimplementovat pro něj systém pro doporučování příspěvků. K hledání souvisejících příspěvků byl využit algoritmus PageRank s využitím vah vztahů mezi příspěvky. Pro jeho využití jsou data uložena v podobě grafu do grafové databáze Neo4j.

Pro doporučení získaných souvisejících článků byl vyvinut modul pro redakční systém Drupal. Součástí modulu je *blok*, který za pomoci PHP Neo4j driveru získává související články z grafové databáze a vykresluje je do HTML pomocí nástroje Twig.

Řešení je přínosné pro návštěvníky webu, kterým zjednoduší orientaci na webu Fakulty informačních technologií ČVUT v Praze. Pomocí doporučených článků bude mít návštěvník možnost dostat se k souvisejícím článkům rychleji bez dlouhého procházení hierarchie článků. Zároveň řešení pomůže tvůrcům webu, kterým ubyde práce s manuálním přiřazováním doporučených článků v administraci Drupalu.

Ačkoliv výsledné řešení poskytuje relevantní výsledky, existuje zde prostor pro budoucí práci, která by přinesla další zkvalitnění výsledných doporučení pomocí hlubší analýzy významů vztahů mezi příspěvky a úpravou jejich vah použitých ve výpočtu PageRanku.

Seznam literatury

1. FIELDING, Roy T.; RESCHKE, Julian. *Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content* [online]. RFC Editor, 2014 [cit. 2019-05-14]. Č. 7231. Dostupné z DOI: 10.17487/RFC7231.
2. DRUPAL ASSOCIATION. *Overview* [online]. 2018 [cit. 2019-05-13]. Dostupné z: https://www.drupal.org/docs/8/understanding-drupal-8/overview.
3. LLC, Google. *Overview | Analytics Real Time Reporting API* [online]. 2016 [cit. 2019-05-14]. Dostupné z: https://developers.google.com/analytics/devguides/reporting/realtime/v3/.
4. *Anonymous Browser Fingerprinting* [online]. 2013 [cit. 2019-05-13]. Dostupné z: http://valve.github.io/blog/2013/07/14/anonymous-browser-fingerprinting/.
5. ROBINSON, Ian; WEBBER, Jim; EIFREM, Emil. *Graph Databases: New Opportunities for Connected Data*. 2. vyd. O'Reilly Media, Inc., 2015. ISBN 9781491930892. Dostupné také z: https://graphdatabases.com/.
6. NEO4J, Inc. *Relational Databases vs. Graph Databases: A Comparison* [online] [cit. 2019-05-13]. Dostupné z: https://neo4j.com/developer/graph-db-vs-rdbms/.
7. VICKNAIR, Chad; MACIAS, Michael; ZHAO, Zhendong; NAN, Xiaofei; CHEN, Yixin; WILKINS, Dawn. A comparison of a graph database and a relational database: A data provenance perspective. *The 48th ACM Southeast Conference* [online]. 2010 [cit. 2019-05-14]. Dostupné z DOI: 10.1145/1900008.1900067.
8. SONYA LIBERMAN Baruch Brutman, Shahar Fleischman. *Looking at Content Recommendation Through a Search Lens* [online]. 2017 [cit. 2019-04-20]. Dostupné z: https://www.elastic.co/blog/looking-at-content-recommendation-through-a-search-lens.

9. PAGE, Lawrence; BRIN, Sergey; MOTWANI, Rajeev; WINOGRAD, Terry. *The PageRank Citation Ranking: Bringing Order to the Web.* [online]. Stanford InfoLab, 1999 [cit. 2019-04-20]. Č. 1999-66. Dostupné z: `http://ilpubs.stanford.edu:8090/422/`.

10. GLEICH, David F. PageRank beyond the Web. *SIAM Review* [online]. 2014, roč. 57, č. 3 [cit. 2019-05-14]. ISSN 0036-1445. Dostupné z DOI: `10.1137/140976649`.

11. NEO4J, Inc. *The PageRank algorithm* [online] [cit. 2019-05-14]. Dostupné z: `https://neo4j.com/docs/graph-algorithms/current/algorithms/page-rank/`.

12. GUPTA, Pankaj; GOEL, Ashish; LIN, Jimmy; SHARMA, Aneesh; WANG, Dong; ZADEH, Reza. *WTF: The Who to Follow Service at Twitter* [online]. 2013 [cit. 2019-05-14]. Dostupné z: `https://stanford.edu/~rezab/papers/wtf_overview.pdf`.

13. XING, Wenpu; GHORBANI, Ali. Weighted PageRank algorithm. *Proceedings. Second Annual Conference on Communication Networks and Services Research* [online]. 2014 [cit. 2019-05-14]. Dostupné z DOI: `10.1109/DNSR.2004.1344743`.

14. *DB-Engines Ranking of Graph DBMS* [online]. solid IT gmbh, 2019 [cit. 2019-05-13]. Dostupné z: `https://db-engines.com/en/ranking/graph+dbms`.

15. *Method of calculating the scores of the DB-Engines Ranking* [online]. solid IT gmbh, 2019 [cit. 2019-05-13]. Dostupné z: `https://db-engines.com/en/ranking_definition`.

16. PAGE, Lawrence; BRIN, Sergey. *Seventh International World-Wide Web Conference (WWW 1998).* The Anatomy of a Large-Scale Hypertextual Web Search Engine [online]. 1998 [cit. 2019-05-14]. Dostupné z: `http://ilpubs.stanford.edu:8090/361/`.

17. DRUPAL ASSOCIATION. *About nodes* [online]. 2017 [cit. 2019-05-13]. Dostupné z: `https://www.drupal.org/docs/8/core/modules/node/about-nodes`.

18. NEO4J, Inc. *Cypher projection* [online] [cit. 2019-05-14]. Dostupné z: `https://neo4j.com/docs/graph-algorithms/current/projected-graph-model/cypher-projection/`.

Dodatek A

Seznam použitých zkratek

AJAX Asynchronous JavaScript and XML

API Application Programming Interface

ASCII American Standard Code for Information Interchange

HTML Hypertext Markup Language

HTTP Hypertext Transfer Protocol

NID Node identifier

SQL Structured Query Language

URL Uniform Resource Locator

Dodatek B

Obsah přiloženého USB flash disku

- `readme.txt` ..... stručný popis obsahu USB flash disku
- `thesis.pdf` ..... text práce ve formátu PDF
- `fit-recommendation-block.tar.gz` ..... zaarchivovaný modul FIT recommendation block
- `src`
  - `fit-recommendation-block` ..... zdrojové kódy implementace
  - `thesis` ..... zdrojová forma práce ve formátu LaTeX