FAKULTA APLIKOVANÝCH VĚD
ZÁPADOČESKÉ UNIVERZITY
V PLZNI

<KIV> KATEDRA INFORMATIKY
A VÝPOČETNÍ TECHNIKY

IR - Semestrální práce
Komplexní IR systém

Západočeská Univerzita V Plzni
Katedra Informatiky a výpočetní techniky
Semestrální práce - IR

Vladimíra Kimlová
letní semestr
11. května 2024

Obsah

1 Zadání

Implementace vlastního systému automatické indexace a vyhledávání dokumentů.
Minimální nutná funkčnost semestrální práce:
Tokenizace, preprocessing (stopwords remover, stemmer/lemmatizer), vytvoření in-memory invertovaného indexu, tf-idf model, cosine similarity, vyhledávání dotazem vrací top x výsledků seřazených dle relevance (tj. vektorový model - vector space model), vyhledávání logickými operátory AND, OR, NOT (booleovský model), podrobná dokumentace (programátorská i uživatelská), podpora závorek pro vynucení priority operátorů.
Semestrální práce **musí umožňovat** zaindexování dat stažených na cvičení (1. bodované cvičení Crawler) a libovolných dalších dat ve stejném formátu. Obě sady dat je možné zaindexovat nezávisle na sobě.
Semestrální práce **musí umožňovat** zadávat dotazy z GUI nebo CLI (command line interface) a při zadávání dotazů je možno vybrat index a model vyhledávání (vector space model vs. boolean model). Výsledky vyhledávání **obsahují i celkový počet dokumentů**, které odpovídají zadanému dotazu.
Semestrální práce **musí umožňovat** zaindexování evaluačních dat, vyhledávání v nich a spuštění evaluace. Pokud pracujete pouze s daty v Angličtině, není evaluace povinná (jelikož nemáme evaluační data pro Angličtinu)

Nadstandardní funkčnost(lze získat až dalších 15 bodů)

- File-based index
- pozdější doindexování dat - přidání nových dat do existujícího indexu
- ošetření HTML tagů
- detekce jazyka dotazu a indexovaných dokumentů
- vylepšení vyhledávání
- vyhledávání frází (i stop slova)
- vyhledávání v okolí slova
- více scoring modelů
- indexování webového obsahu - zadám web, program stáhne data a rovnou je zaindexuje do existujícího indexu (2b)
- další předzpracování normalizace

- GUI/webové rozhraní
- napovídání keywords
- podpora více polí pro dokument (např. datum, od do)
- zvýraznění hledaného textu v náhledu výsledků
- vlastní implementace parsování dotazů bez použití externí knihovny
- implementace dalšího modelu (použití sémantických prostorů, doc2vec, Transformers - BERT) atd. (1-5b podle náročnosti)
- podpora více jazyků
- jakékoliv další užitečné vylepšení, které vás napadne

2 Implementace

Implementace celého IR systému byla provedena v jazyce *Python*, byly využity standardní knihovny *numpy* a *pandas*, dále byla využita knihovna *lxml* při stahování obsahu webových stránek. Pro uživatelské rozhraní byla použita knihovna *PyQt5*. Pro lemmatizaci bylo využito knihovny *simplemma*.

2.1 Web Crawler

V rámci semestrální práce byla nacrawlována česká webová stránka `https://theelderscrolls.fandom.com`. Samotné crawlování lze provést pro jednu webovou stánku danou její URL adresou nebo lze zadat „seedovací“ stránku a budou staženy všechny stránky, na které bude v průběhu crawlování odkazováno. Příslušná data jsou uložena do složky. Bylo nacrawlováno 855 dokumentů přičemž většina z nich je psána česky, ale část dokumentů je ve slovenštině.

2.2 Předzpracování dat

Vzhledem k povaze stažených dat probíhalo celé předzpracování jak pro český, tak pro slovenský jazyk. Pro předzpracování byly navrženy 3 základní pipeliny: stemovací, lemmatizovací a tokenizovací. V rámci tokenizace je vstupní text převeden na malá písmena a jsou odstraněny HTML značky, citační značky (např. [1]) a závorky obecně. Dále jsou volitelně odstranitelná i *stopslova*[1], jak česká, tak slovenská. Text je rozdělen

[1] `https://github.com/stopwords-iso`

podle mezer, webové odkazy, datumy, emailové adresy a slova obsahující * jsou brány jako jeden token. Obecně je také tokenizací odstraněna interpunkce, ale pro tvorbu snippetu je i možnost tokenizace zachovávající interpunkci.
Lemmatizovací pipeline je pak kombinací lemmatizace a tokenizační pipeliny. Pro lemmatizaci byla využita lightweight knihovna *simplemma*, která nativně umožňuje lemmatizaci pro češtinu i slovenštinu zároveň. Byly testovány i jiné knihovny např. v evaluaci bude dále zmíněna knihovna *lemmagen3*, osobně mi přišlo (a evaluace to potvrdila), že výsledky byly obecně horší. Existuje i mnoho lemmatizerů, které často vyžadovaly stažení modelu či jiné složitější konfigurace, opět dle subjektivního názoru byly v tomto případě výsledky lepší, bohužel však tato řešení vzhledem k množství dat nebyla uskutečnitelná, protože by pak předzpracování trvalo neúnosně dlouho.
Stemovací pipeline je pak kombinací stemmingu a tokenizační pipeline. Pro stematizaci se mi nepodařilo nalézt žádnou knihovní implementaci zahrnující jak český, tak slovenský jazyk. Implementace stemmeru byla proto provedena ručně na základně inspirace řešeními z GitHubu[2,3]. Stematizace je provedena buď pro češtinu, nebo slovenštinu dle rozpoznaného jazyka. Fungování stemmeru je dobře vidět na Obrázku 3.

2.3 Indexace

Indexace dokumentů je zajištěna třídou `Indexer`, vytvořením více instancí této třídy lze vytvořit i více indexů nad libovolnými daty. Je počítáno s pevně danou strukturou indexovaných dat danou webovým crawlerem, je předpokládáno, že dokumenty se člení na: nadpis, obsah, tabulku a hlavní text.
Index lze obecně vytvořit ze složky s nacrawlovanými dokumenty nebo jej vytvořit na základě seedovací URL (pak jsou data nejprve stažena). V rámci indexace je obecně vytvořeno několik struktur; cache dokumentů, invertovaný index, normy dokumentů a klíčová slova. Samotná indexace pak začíná vytvořením cache dokumentů, zde jsou uloženy jednotlivé části dokumentů, interně přidělené ID a detekovaný jazyk dokumentů – pro detekci jazyka byl použit „všejazyčný“ model ze cvičení, navíc byl ještě vytvořen pouze česko-slovenský model, který mi přijde vzhledem k povaze dokumentů relevantnější. Dále jsou zde uloženy pomocné struktury: maximální ID a nepoužitá ID (ID v rozsahu 0 až max ID, která jsou volná, tj. ID smazaných dokumentů). Dále jsou jednotlivé dokumenty předzpracovány, tedy převedeny na relevantní termy, na základě těchto termů je tvořen invertovaný index a normy dokumentů. V rámci celého indexu jsou uloženy: frekvence dokumentů, inverzní

[2] `https://research.variancia.com/czech_stemmer/`
[3] `https://github.com/mrshu/stemm-sk/tree/master?tab=readme-ov-file`

dokumentová frekvence, norma, mapa ID dokumentů a k nim jejich frekvence termů a tf-idf skóre a také poziční index. Takto se může zdát, že jsou zbytečně ukládány mezivýsledky. Předpočtené tf-idf skóre a normy jsou uloženy za účelem urychlení vyhledávání, naopak mezivýpočty jsou uloženy pro jednodušší přepočtení při změně dokumentů v indexu (smazání, úprava, vytvoření). Například nebylo-li by uloženo tf, bylo by obtížné přepočítat tf-idf dokumentu v případě, že idf bylo 0 – tj. daný term byl pouze v jednom dokumentu. Po přenásobení 0 by informace o tf v rámci tf-idf zanikla a změnilo-li by se pouze idf (např. přidáním nového dokumentu), muselo by být tf znovu napočítáno. Při tvorbě indexu byla obecně snaha o efektivitu, není tedy například tvořena celá váhová matice, ale bylo pracováno pouze s relevantními (nenulovými) prvky.

Jak již bylo nastíněno, jsou kromě samotné tvorby indexu umožněny i jeho modifikace, tj. přidání nového dokumentu (ze souboru nebo na základě URL), upravení části dokumentu, smazání dokumentu. Všechny tyto operace byly navrženy tak, aby se zásah do indexu projevil pouze na relevantních pozicích. To znamená, že v případě zásahu do indexu není provedena celá indexace nanovo, nýbrž jsou přepočteny pouze ovlivněné záznamy. Například při odstranění dokumentu dochází zjednodušeně k odstranění záznamů týkajících se tohoto dokumentu (např. norma) a přepočtení skóre pouze těch dokumentů, co s odstraněným dokumentem sdílejí term.

2.4 Vyhledávání

Při vyhledávání je dle zadání možno zadat index, v němž se má vyhledávat, model vyhledávání (vektorový nebo booleovský model), sekce, v níž se má vyhledávat (nadpis, obsah, tabulka, hlavní text, celý dokument), a počet výsledků k zobrazení. Vyhledané nejlepší dokumenty jsou zobrazeny v GUI společně se snippety. Snippety obecně fungují tak, že zahrnují nejlepší nalezené okénko dané velikosti, které obsahuje co nejvíce slov z dotazu (a trochu kontextu okolo), výsledky vyhledávání pro jeden konkrétní dotaz jsou vidět na Obrázku 1. Snippety jsou tvořeny na základě pozičního indexu, je proto potřeba, aby byla zaindexována i stopslova.

2.4.1 Vektorový model

Vektorový model využívá napočtená tf-idf skóre a normy. Pro vstupní dotaz (který je předzpracován stejnou pipeline jako indexované dokumenty) jsou tyto hodnoty také napočeny a je určena výsledná kosinovská podobnost. Na základě těchto podobností je vybráno `k` nejlepších výsledků a pro tyto dokumenty jsou vytvořeny snippety. Při vyhledávání v celém dokumentu jsou napočteny podobnosti nejprve v rámci

jednotlivých sekcí a následně zkombinovány takto:

$$nadpis\_skóre * 1.1 + obsah\_skóre + tabulka\_skóre * 0.5 + hlavní\_text\_skóre * 0.5. \quad (1)$$

Na základě těchto zkombinovaných podobností jsou pak vybrány nejlepší dokumenty.
Speciálním případem vyhledávání ve vektorovém modelu je pak vyhledávání v okolí slova (proximity search), respektive vyhledávání frází (řešeno jako vyhledávání s proximitou 1). Tato situace je řešena jako postfiltrování klasického vyhledávání. To znamená, že dotaz je vyhledán bez ohledu na blízkost jednotlivých slov v dotazu, takto nalezené dokumenty jsou posléze redukovány na základě (ne)splnění podmínky blízkosti dle pozičních indexů. Proximity vyhledávání je prováděno pouze nad hlavním textem dokumentů. Tvorba snippetů v tomto speciálním případě nebere okénko maximalizující počet slov, ale okénko obsahující danou frázi či slova s danou vzdáleností – velikost zde tedy záleží na povaze dotazu, snippety pro tuto situaci jsou k nahlédnutí na Obrázcích 2 a 3.

2.4.2 Booleovský model

Pro správnou funkcionalitu booleovského modelu bylo nutno naimplementovat parser booleovských dotazů, ten funguje na principu převodníku z infixové do postfixové notace. Implicitně je mezi tokeny (opět předzpracovány stejnou pipeline jako indexované dokumenty) v dotazu doplňován OR, precedence operátorů je pak v tomto pořadí sestupně: NOT, AND, OR. Booleovský model využívá seznamy ID dokumentů z inverovaného indexu pro dané termy, pro každý term z dotazu je tedy takto získáno pole ID dokumentů, v nichž se vyskytuje. Vyhodnocení jednotlivých logických operací je pak prováděno pomocí množinových operací. V případě operátoru NOT je vytvořena množina všech ID bez ID dokumentů s daným termem. Ukázka booleovského vyhledávání je na Obrázku 1.
Samotná implementace booleovského vyhledání se snaží být poměrně robustní ve smyslu vyhledání. Tímto je myšleno, že je snaha vrátit výsledky i pro nevalidní booleovské dotazy. Následuje několik příkladů:

- `skyrim AND ,` je vyhodnoceno jako `skyrim`
- `železná AND OR dýka` je vyhodnoceno jako `železná OR dýka`
- `AND železná dýka` je vyhodnoceno jako `železná dýka`

2.5 Shrnutí implementovaných vylepšení

- Modifikace indexu: pozdější doindexování dat, úprava dokumentu, smazání dokumentu – přepočet pouze relevantních záznamů (výrazně rychlejší než celá indexace)
- Ošetření HTML tagů
- Detekce jazyka dotazu a indexovaných dokumentů – na základě detekovaného jazyka je provedeno předzpracování daného dokumentu
- Vylepšení vyhledávání – kliknutím na nadpis je možno zobrazit celý hlavní text dokumentu
- Vyhledávání frází (i stop slova)
- Vyhledávání v okolí slova
- Indexování webového obsahu – pouze pro webové stránky v daném formátu
- Další předzpracování normalizace – dle detekovaného jazyka, vlastní stemmer a tokenizer
- GUI
- Napovídání keywords
- Podpora více polí pro dokument
- Zvýraznění hledaného textu v náhledu výsledků – vhodné okénko maximalizující počet slov z dotazu, pro proximity vyhledávání dynamické okénko zahrnující nalezená slova v dané vzdálenosti
- Vlastní implementace parsování dotazů bez použití externí knihovny
- Podpora více jazyků – čeština a slovenština

3 Evaluace

Vzhledem k tomu, že indexovaná data byla (z většiny) česky, byla provedena také evaluace na dodaných evaluačních datech. Zde se vyskytl určitý problém v integraci, neboť předložená data byla poskytnuta ve formátu Javovského binárního souboru, což vyžadovalo jejich převod do formátu zpracovatelného v jazyce Python. Další komplikací byl pak formát výstupních dat, která sloužila jako vstup pro přiložený

evaluační program. Vzhledem k absenci oficiálního popisu jsem byla nucena odvodit tento formát z přiloženého kódu napsaného v jazyce Java.
Výsledky samotné evaluace jsou pak v Tabulce 1. Bylo testováno několik různých kombinací, porovnání stemmeru a lemmatizeru, indexace včetně a bez stopslov a dále povaha vstupního dotazu.

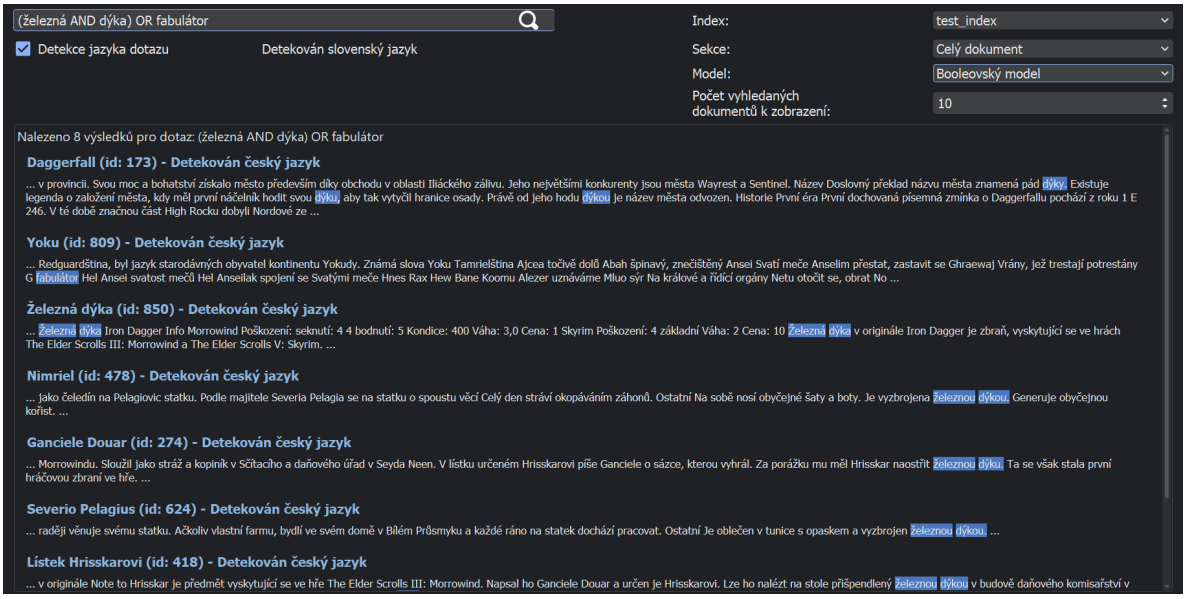
Tabulka 1: Výsledné hodnoty MAP na evaluačních datech

Metoda	**MAP**
Stematizace	
pouze nadpis	0.0895
nadpis i popis	0.0997
pouze popis	0.0684
pouze nadpis, včetně stopslov	0.0864
Lemmatizace – Simplemma	
pouze nadpis	0.0283
nadpis i popis	0.0099
pouze popis	0.0102
pouze nadpis, včetně stopslov	0.0224
Lemmatizace – Lemmagen3	
pouze nadpis	0.0176

Nejlepších hodnot MAP skóre bylo překvapivě dosaženo při použití stemmeru a pro kombinaci jak nadpisu, tak popisu pro dotazy, konkrétně bylo dosaženo 10 %. Výrazně horší lemmatizace oproti stematizaci může být způsobena tím, že byly zvoleny odlehčené verze lemmatizovacích knihoven (které ale byly rychlostně výkonné).

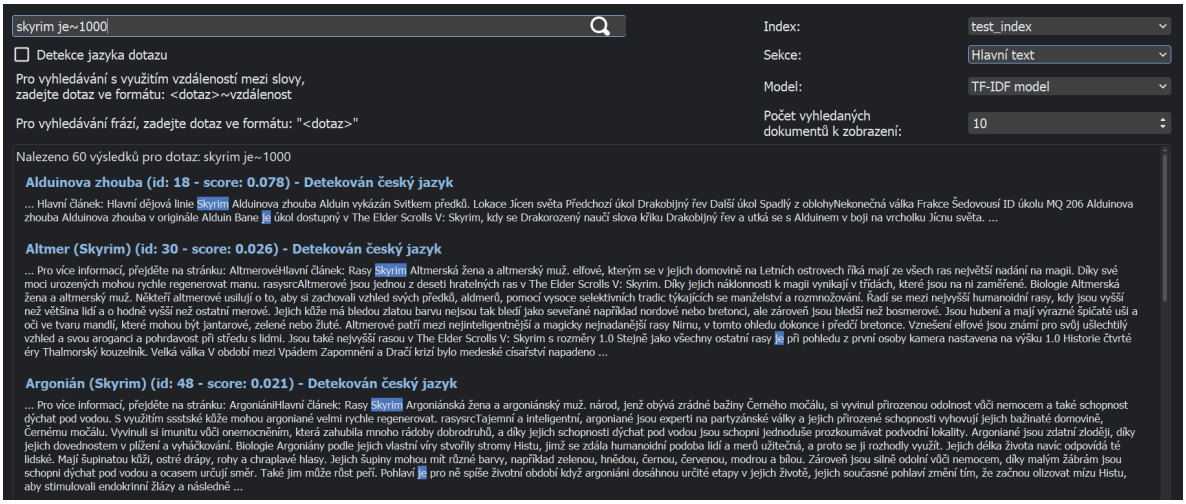
4 Uživatelská příručka a uživatelské rozhraní

Jak již bylo zmíněno celá aplikace byla vyvíjena v jazyce Python, konkrétně je doporučena veze 3.9.7. Pro stažení potřebných Python modulů je v rámci aplikace k dispozici soubor `requirements.txt`. Hlavním spustitelným souborem je pak soubor `searcher_gui.py`, který spustí uživatelské rozhraní pro vyhledávač. Před jeho spuštěním je nutno v souboru `config.py` vytvořit indexy, v nichž se bude vyhledávat. Zároveň byl přiložen soubor `demo.ipynb`, který demonstruje možné operace nad indexy (vytvoření, přidaní dokumentu, ...).

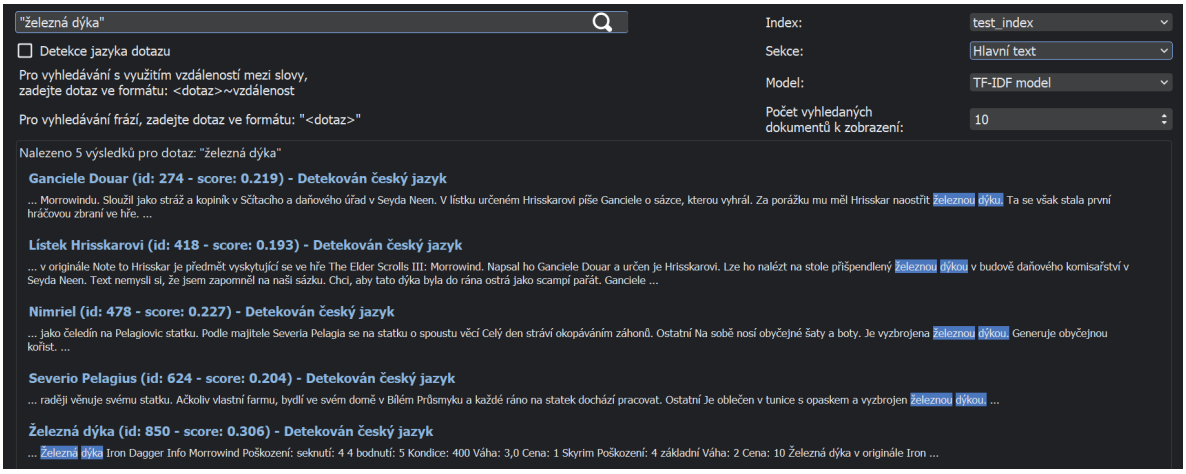


Obrázek 1: Uživatelské rozhraní – vyhledání dotazu `(železná AND dýka) OR fabulátor` booleovským modelem

Grafické uživatelské rozhraní pak slouží jako samotný vyhledávač (search engine). V horní části se nachází vyhledávací panel, do nějž lze zadávat (formátované) dotazy. Pod vyhledávacím panelem lze zaškrtnout detekci jazyka dotazu. V pravé horní části se pak nachází další nastavení vyhledávání, konkrétně výběr z indexu (indexy nutno vytvořit v `config.py`), výběr sekce, v níž se bude vyhledávat, výběr modelu vyhledávání a výběr počtu výsledků k zobrazení.
Pro booleovské vyhledávání je nutno používat pouze podporované operátory, tj. AND, OR, NOT (přičemž OR je implicitní), nebo závorky. Jakékoliv jiné operátory budou vnímány jako tokeny pro vyhledání. Příklad je vidět na Obrázku 1.
Při vyhledávání vektorovým modelem může být obecně dotaz v jakémkoliv formátu. Pevně daný formát je pouze pro vyhledávání v blízkosti a pro vyhledávání frází. Pro vyhledávání frází je nutno dotaz obalit do uvozovek. Vyhledávání v blízkosti je pak vázáno na znak $\sim$, za nímž je očekávána hodnota blízkosti, tj. číslo. Příklady použití jsou na Obrázcích 2 a 3.



Obrázek 2: Uživatelské rozhraní – vyhledání dotazu s proximitou vektorovým modelem



Obrázek 3: Uživatelské rozhraní – vyhledání fráze vektorovým modelem

5 Závěr

V rámci semestrální práce byl vypracován komplexní IR systém, který slouží pro vyhledávání v datech ze stránky `https://theelderscrolls.fandom.com` či jiné stránky stejného formátu. Vyhledávač mimo jiné umožňuje hledání v blízkosti slova či podporu češtiny i slovenštiny. V rámci evaluace bylo zjištěno, že nejvhodnější je využití stemmeru, což mi osobně přijde pro český jazyk překvapivé, intuitivně bych čekala lepší výsledky od lemmatizace.

Samotná evaluace byla trochu krkolomná tím, že byla integrována pouze pro Javu. Špatné výsledky evaluace mohou proto být způsobeny špatným přeparsováním dat z Javovské binárky nebo nesprávným formátem výstupního souboru evaluace.

Do budoucna by se dala aplikace rozšířit o podporu dalších jazyků a hlavně dalších formátů vstupních indexovaných dat. Dále by se daly také zavést další skórovací modely – například založené na neuronových sítích.