Semestrální práce z KIV/PT

Patrik Harag
Michal Fiala

29.11.2016

Obsah

Semestrální práce z KIV/PT

Zadání

Vytvořte program, který načte text ze souboru a ze všech unikátních slov vyskytujících se v textu vytvoří slovník. Slovník bude možné vyexportovat ve vhodně zvoleném formátu do souboru a znovu použít při dalším spuštění programu. Program dále umožní napsat nebo načíst libovolný text a v něm vyhledat zadané slovo. Pokud se slovo bude vyskytovat v prohledávaném textu, tak program vypíše počet výskytů a uvede u všech výskytů počáteční a koncový index, kde se v textu slovo nachází. Pro prohledávání textu využijte algoritmu komprimované trie. Pokud se zadané slovo v textu nenachází, tak vypište maximálně 10 nejbližších slov ze slovníku. K porovnání vzdáleností mezi hledaným slovem a slovy ze slovníku použijte Levensteinovu metriku probíranou na přednáškách. Uživatel by měl mít také možnost přidat hledané slovo do slovníku, pokud se v něm nenachází.

Uživatelské rozhraní programu může být grafické i konzolové. Program bude mít ošetřeny všechny vstupy a zdrojový kód projde validací nástrojem PMD. Kód programu bude okomentovaný javadoc komentáři.

Analýza problému

Slovník jsme se rozhodli implementovat pomocí hašovací tabulky. To umožní testování výskytu slova v konstantním čase. Export slovníku lze velice snadno vyřešit uložením všech slov do textového souboru, každé slovo na jeden řádek.

Vyhledávání v textu bude realizováno pomocí komprimované trie. Ta se vytvoří ze zadaného textu, který je pomocí třídy `Pattern` rozdělen do slov. Tyto rozdělená slova po jednom bude vkládat i s jejich indexy v textu do metody pro vkládání do standardní trie. V metodě bude probíhat kontrola zda-li slovo již nebylo vloženo a nebo zda-li nebyl vložen jeho prefix. Pokud nebylo slovo vloženo nebo byl vložen jeho prefix vytvoří nové uzly odpovídající slovu. Po vložení se na posledním vloženém uzlu vloží (uloží) indexy slova v textu. Tento proces se bude opakovat dokud nebudou vložená všechna nalezená slova. Poté, co bude vytvořena standardní trie, bude použita metoda pro upravení této trie na komprimovanou. Tato metoda projde veškeré uzly stromu a pokud uzel nebude koncem slova a bude mít pouze jednoho následovníka spojí se jejich hodnoty a napojí se jejich následovníci.

Vyhledávání bude fungovat na bázi prohledávání každé úrovně stromu od kořene stromu. Hledané slovo se bude porovnávat s následovníky kořene a pokud bude nalezen následovník, který je prefixem nebo slovem, přejde se na tohoto následovníka a pokud bylo slovo prefixem bude pokračovat hledání se zbylou částí slova, pokud byl následovník slovem(zbylou částí slova) vloží se do pole indexy slova v textu. Pole bude prázdné pokud hledané slovo v trii není vytvořené.

Během veškeré práce s textem budou všechny znaky převedeny na malé. Při rozdělování textu na jednotlivá slova je nutné dávat pozor na nepísmenné znaky. Kromě očekávané tečky a čárky se v textu mohou třeba vyskytovat uvozovky nebo pomlčky. Funkčnost aplikace musí být řádně otestována také pro řetězce s diakritikou.

Návrh programu

V programu je řádně oddělena definice uživatelského rozhraní (balíček `cz.zcu.kiv.pt.sp.app`) od logiky (balíček `cz.zcu.kiv.pt.sp`).

Uživatelské rozhraní

Pro vytvoření uživatelského rozhraní byl využit framework JavaFX. Celé uživatelské rozhraní je definované pomocí FXML (`FXMLDocument.fxml`) a obsluhované kontrolerem (`FXMLDocumentController.java`).

Slovník

Rozhraní definující slovník se nazývá `Dictionary`, jeho jediná implementace je `HashSetDictionary`, která využívá `HashSet` z Java Collections API. Třída `DictionaryCreator` poskytuje metody pro vytváření slovníku z textu a třída `DictionaryIO` metody pro načítání a ukládání slovníku do souboru.

Vyhledávání v textu

Rozhraní pro třídu umožňující vyhledávání se nazývá `SearchProvider`. Implementacemi tohoto rozhraní jsou `RegexSearch`, který vyhledává slova v textu pomocí regulárních výrazů, a `TrieSearch`, který vyhledává pomocí komrimované trie.

Levenshteinova vzdálenost

Výpočet Levenshteinovy vzdálenosti je implementován ve třídě `LevenshteinDistance`.

Testování

Pro ověření funkčnosti byly vytvořeny jednotkové testy pomocí JUnit 4.

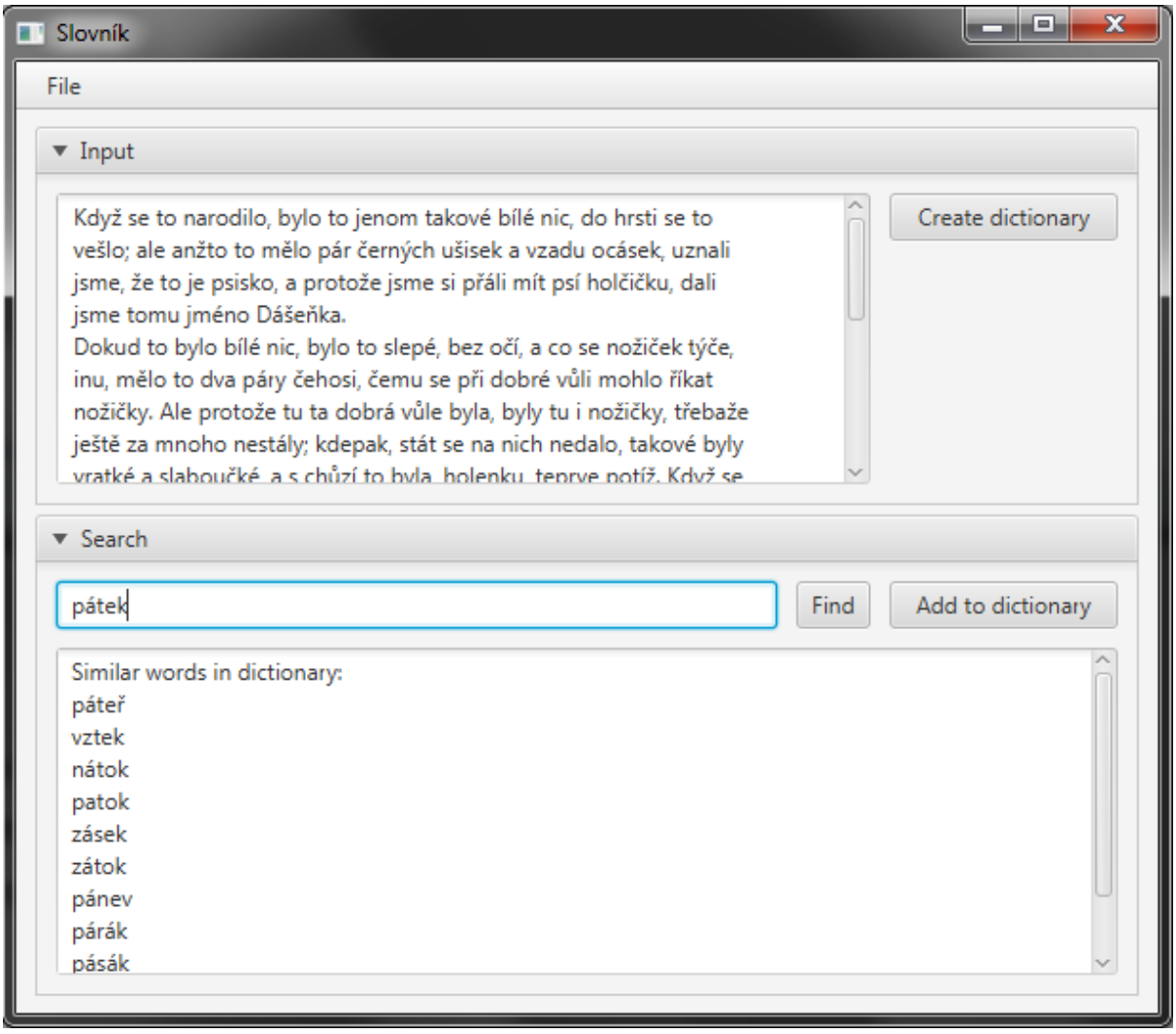
Uživatelská dokumentace

Ke spuštění je vyžadovaná Java 8.

Po spuštění programu se zobrazí grafické uživatelské rozhraní. Uživatelské rozhraní je rozděleno na dvě části. První část slouží k zadání vstupu – obsahuje textové pole a tlačítko pro vytvoření slovníku z textu v tomto poli. Druhá část umožňuje vyhledávání v zadaném textu a obsahuje také pole pro zobrazení výsledků. Vyhledávané slovo je případně možné uložit kliknutím na příslušné tlačítko.

Aplikace dále obsahuje menu, které umožňuje následující akce:

- Otevřít libovolný textový soubor a jeho obsah vložit do vstupného textového pole (kódování textu musí být UTF-8)
- Načíst slovník vytvořený aplikací při dřívějším použití, případně vytvořený uživatelem nějakým jiným způsobem (na jedné řádce jedno slovo, kódování UTF-8)
- Uložení slovníku do souboru.
- Uložení textového výstupu vyhledávání do souboru.
- Ukončení programu.



Závěr

Tento projekt nám především ukázal jaké plusy a mínusy přináší práce v týmu. Komprimovaná trie a vyhledávání nám funguje bez problémů. Avšak je na zvážení pokud ušetří čas oproti předchozímu řešení pomocí regulárních výrazů. Program funguje dle požadavků zadání. Bez problémů si poradí i s velkými slovníky.