Popis řešení

Shrnutí

Během řešení jsem používal Perl poprvé, takže jsem musel mnoho vyhledávat. A i proto jsem se rozhodl problém hledání klíčových slov řešit jednoduše možná by se dalo říci až primitivně.

Vycházím z toho, že klíčoá slova by pravděpodobně měla patřit mezi ta nejvíce používaná. Daný text tedy postupně procházím po slovech a pomocí asociativního seznamu počítám výskyty jednotlivých slov.

Nakonec iteruji seznamem klíčů (tedy slov) seřazených právě podle jejich četnosti (tedy hodnoty pod klíčem v asoc. seznamu) a vypíšu pouze několik nejfrekventovanějších.

V průběhu implementace jsem se rozhodl umožnit některé parametry nastavit pomocí přepínačů, i když to nebylo v zadání. Skript je tak mnohem užitečnější.

Detailnější popis

Především na začátku skriptu zparsuji argumenty a nastavím si podle toho proměnné - subprocedura **`parse_args`**. Určitě existují funkce, které zpracují argumenty podle standardu a které by mi dovolili zpracovat sofistikovanější syntax přepínačů, ale já jsem pro takovýto jednoduchý skript zůstal u "ručního" regex porovnávání a klasické rovnosti.

Poté si načtu slovník slov, které chci ignorovat v **`load_ignores`**. To dělám tak, že se pokusím otevřít určený soubor a pokud neexistuje tak se vrátím a nechám **`%ignore`** hash prázdný. Pokud se mi podaří soubor otevřít, tak z každéh řádku odstraním newline znak a použiji ho jako klíč v **`%ignore`**.

Následně přichází o něco méně elegantní větvení. Nepodařilo se mi totiž zjistit jak file handly uložit do proměnné, a tak dochází k nepěkné duplikaci kódu. Pokud totiž byla předána cesta k souboru, nejdřív soubor otevřu (v režimu s UTF8 kódováním), pokud ne tak pouze nastavím standardnímu vstupu UTF8 kódování, a pak v obou větvích iteruji po řádcích, které rozděluji pomocí **`split`** na slova. Pokud je slovo kratší než požadovaná délka (daná buď přepínačem nebo výchozí hodnotou), tak jej přeskočím. Jinak slovo v malých písmenech použiji jako klíč a zvýším hodnotu.

Dále pokud byl použit přepínač **`-pN`** nebo nebyl pužit přepínač **`-cN`** najdu **`N`**. respektive výchozí 95. percentil výskytů a tuto hodnotu použiji jako hranici výskytu.

Pak už jenom vypíši na standardní výstup všechna slova - tedy klíče z **`%key_words`** - v pořadí od nejčetnějších po méně četná, přičemž přeskakuji slova, která jsou klíčem v **`%ignore`**, a skončím vypisovat, pokud narazím na slovo s menší hodnotou než je hranice.

Reflexe a komentář

Největším nedostatkem je, že slova jako předložky a spojky se často v textu opakují více než klíčová slova relevantní pro předmět textu. To by jistě nemělo množinu podobných dokumentů zmenšit, ale potenciálně by se tak mohly zdát podobné dokumenty, které si nijak příbuzné nejsou. Tento problém částečně řeší možnost přidat soubor se slovy, která nechceme brát jako klíčová, ale pak samozřejmě záleží na jeho rozsáhlosti.

Pak také úkol stěžuje skloňování slov, které v řešení ignoruji. Jistě by se dalo pracovat detailněji s kořeny slov, ale jejich algoritmycké hledání by bylo nesnadné.

Zároveň jsem skript testoval pouze pro krátké a středně dlouhé texty, kde samozřejmě čím delší text je tím je větší šance že mezi nejfrekventovanějšími slovy budou skutečně taková, která mají nějaký vztah k předmětu textu.

Nebyl jsem si jistý přirozeným jazykem, který jsem měl použít. Zadání požaduje, aby se soubor jmenoval česky a i uživatelský manuál jsem napsal v češtině. Jsem ale více zvyklý psát názvy a komentáře v angličtině a obecně je dle mého angličtina vhodnějším jazykem v kontextu kódu. A jelikož jsem už v kódu měl angličtinu použil jsem ji i pro výstup při chybových hláškách a přepínače help. Uznávám tedy tuto jazykovou dualitu a jistě bych neměl problém psát kód česky nebo naopak manuál napsat v angličtině.

A i v rámci stylu není kód 100% konzistentní, protože jsem se Perl učil za běhu, přebíral některé zvykolsti, které jsem pochytil, a přirozeně je kombinoval se svými zvyky a preferencemi. Stejně tady bych neměl problém se naučit stylu, který je preferovaný v týmu. V jazycích, které znám lépe píši kód čistěji.

Na závěr bych zmínil pár možných rozšíření: přepínač na vypnutí ignorovacího souboru, přepínač na zadání vlastní cesty k ignorovacímu souboru a možnost ztotožnit více tavrů jednoho slova podle přidaného slovníku přípon.