Extrakce českých klíčových slov v jazyce Perl

Tadeáš Jůn

Listopad 2021

1 Úvod

Extrakce klíčových slov je relativně rozvinutý obor - používá se několik hlavních algoritmů, primárně ze dvou kategorií: s či bez využití machine learning. Machine learning metody využívájí velkou databázi trénovacích dat, na kterých se umělá inteligence naučí, co jsou v daném typu textu pravděpodobně klíčová slova.

Při řešení tohoto úkolu jsem neměl přístup k žádným trénovacím datům, ani k přibližnému stylu či obsahu dokumentů, který bude program analyzovat. Proto používat ML přístup nebylo možné, a vybral jsem postup alternativních algoritmů.

Jeden z nejrozšířenějších algoritmů pro extrakci klíčových slov je tf-idf (term frequency–inverse document frequency), který extrahuje slova z jednoho dokumentu z dané kolekce dokumentů. Algoritmus porovnává frekvenci slova v dokumentu na který se zaměřuje, a jeho frekvenci ve všech ostatních přístupných dokumentech. Následně pomocí jednoho ze známých matematických vzorců vypočítá důležitost slova.

Při řešení tohoto úkolu jsem využil upravenou verzi algoritmu tf-idf, která místo porovnávaní frekvence slova v dokumentu s jeho frekvencí ve zbytku databáze ji porovnává s frekvencí v celém českém korpusu.

V této dokumentaci krátce rozvedu teoretický postup extrakce klíčových slov z jednoho dokumentu, praktické řešení problému v jazyce Perl (v5.30.0), uvedu několik příkladů s výsledky a proberu možnosti zlepšení kódu.

Anglická verze této dokumentace je k nalezení v */documentation/eng/documentation.pdf*.

2 Teoretické řešení

V této sekci krátce popíši jak program funguje na teoretické bázi, včetně úpravy textu, práce s korpusem, a přidělení důležitostní hodnoty každému z vybraných slov.

2.1 Zpracování textu

Algoritmus tf-idf by teoreticky měl fungovat sám o sobě, bez žádné výrazné úpravy databáze slov z dokumentu. Nicméně, pro účely tohoto úkolu, zpracování textu a snížení počtu slov, které mají šanci být důležité, výrazně zkracuje čas, který kód potřebuje pro ohodnocení jednotlivých slov. Celé zpracování převede plný český text na jednoduchý seznam zlomku slov z textu. V příkladu 1 v sekci Výsledky zpracování textu převedlo text 11 524 slov na pouhých 345, které poté mohl algoritmus jednotlivě ohodnotit.

Než můžeme začít s zpracováním textu, musíme načíst korpus a samotný dokument, který budeme analyzovat. Korpus českého jazyka je převzaný od Českého Národního Korpusu [1] - jedná se o synchronní reprezentativní srovnávací seznam, který obsahuje více než 100 000 000 textových slov a dat o jejich frekvenci. Samotný dokument poté můžeme načíst ze standardního vstupu, rozdělit na jednotlivá slova, a odstranit interpunkci, závorky, čísla, apod.

První krok zpracování textu je odstranění tzv. stop words. Toto jsou slova, která se v češtině objevují nejčastěji. Jedná se například o slova „a“, „ale“, „na“, „jsem“, atd. Tato slova jednotlivě nepřenáší v textu žádný význam, a proto je můžeme ze seznamu slov vyříznout. Pro tento úkol jsem vybral hranici odstranění 150 nejčastějších slov v češtině - při vyšším limitu tento krok odstraňoval i některá důležitá slova.

Dále je potřeba odstranit krátká slova - slova která mají pouze 3 znaky (nebo méně) obvykle nemají žádný výrazný samostatný význam. Často se jedná o zkratky, či stop words která nejsou v top 150 slovech odstraněných v předchozím kroku.

Následně můžeme pro každé slovo, které zbylo, spočítat jeho frekvenci v analyzovaném dokumentu. Poslední krok zpracování textu je odstranění slov, která se v textu neobjevují moc často - je velmi malá šance, že některé z těchto slov je pro dokument klíčové. Na hranici frekvence, pod kterou jsou slova odstraněna ze seznamu, jsem vybral vzorec

$$\frac{log_e(n)}{log_e(10)} + 0.5$$

kde n je počet analyzovaných slov. Výsledek tohoto vzorce je zaokrouhlený na nejbližší celé číslo, které je následně použito jako zmíněná hranice. Tento přístup způsobuje to, že čím je text delší, tím častější musí být slova aby nebyla v tomto kroku odstraněna - tato škála je však logaritmická, takže hranice u delších dokumentů neroste moc rychle.

Pokud se dané slovo vůbec nevyskytuje v korpusu, je také odstraněno ze seznamu.

Po těchto krocích je text zpracovaný, a je možné přejit na sekci procesu, která přidělí každému slovu, které v seznamu zbylo, důležitostní hodnotu.

2.2 Přidělení důležitnostních hodnot

Jakmile je text zpracován, a seznam slov redukovaný na pouze ty nejdůležitější, může začít proces přidelování hodnot, které indikují odhadovanou důležitost slova pro daný dokument. Algoritmus nyní pro každé slovo ze seznamu provede kalkulaci, která odhaduje tuto hodnotu. Vzorec použitý při počítání hodnot je

$$\frac{x_i}{n_a} \cdot log_e(\frac{\frac{n_d+n_c}{2}}{1+c_i} + 1)$$

kde n_a je počet analyzovaných slov dokumentu, x_i je frekvence daného slova v dokumentu, n_d je součet frekvencí analyzovaných slov, n_c je počet unikátních slov v korpusu (v našem případě 334 833) a c_i je frekvence daného slova v korpusu.

Tyto hodnoty jsou pouze relativní, a proto jsou v dalším kroce normalizovány na rozsah 0.5 - 100 a zaokrouhleny na dvě desetinná čísla, čistě pro zlepšení lidské čitelnosti hodnot.

Celý seznam slov je poté seřazený sestupně podle důležitostních hodnot. Program poté do standardního výstupu vypíše prvních 20 slov (nebo všechna slova, pokud je počet slov menší než 20). V případě, že prográm nemá vybranou možnost simplePrint (viz. Uživatelská dokumentace), program vypíše i důležitostní hodnoty jednotlivých slov.

3 Praktický postup řešení

Program klicova_slova.pl je napsaný v jazyce Perl verze 5.30.0 built for x86_64-linux-gnu-thread-multi. Používá CPAN knihovny Text::CSV_XS::TSV pro čtení korpusu, List::Util pro zpracování textu a Getopt::Long pro načtení uživatelem zadaných parametrů. Obsahuje 8 funkcí, které jsou postupně volány pro zpracování, analýzu a vypsání textu. Následuje krátký popis všech funkcí a celého procesu.

Funkce GetAllWords načítá data po řádcích ze standardního vstupu, následně je rozdělí na jednotlivá slova, která převede do malých písmen a smaže z nich interpunkci a jiné nepotřebné znaky pomocí následujicího seznamu:

```
.,!?;:"'()-[]{}|0123456789
```

Obrázek 1: Seznam znaků odstraněných ze slov.

Jednotlivá slova jsou dále uložena do pole @wordList.

GetCzechCorpus je funkce, která načítá cely český korpus ze souboru syn2015_word_utf8.tsv. Ukládá korpus v poli hashů, které obsahují komponenty rank (pořadí v korpusu), word (tvar slova), a frequency (frekvence slova v celém korpusu).

Následně můžeme začít s zpracováním textu, nejdříve pomocí funkce RemoveStopWords, která odstraní 150 nejčastějších slov v korpusu ze seznamu slov @wordList, a to pomocí dvou vnořených grep příkazů, které filtrují array @wordList pomocí sekundárního pole @frequentCorpusWords.

V dalším kroku, ve funkci RemoveShortWords, se odstraní slova, jejižch délka je menší či rovna 3 znakům.

Funkce GetWordFrequencies počítá frekvence všech zbylých slov v dokumentu, a ukládá je do hashe $\%frequencies$. Jednotlivá slova (keys) v hashi se poté seřadí sestupně podle frekvence, a jejich pořadí se uloží do arraye @sortedKeyFrequencies. Tento seznam se využívá v následujicích sekcích kódu.

V posledním kroků zpracování textu se odstraní málo používaná slova. Hranice je spočítaná vzorečkem zmíněným v sekci Teoretické řešení.

Po úspešném zpracování textu do jeho minimální podoby funkce CalculateImportanceValues spočítá dané důležitnostní hodnoty pro každé zbylé slovo. Tato funkce je pro kód časově nejnáročnější (u delších textů, např. beletrie,

může zabrat i déle než minutu). Po spočítání jsou slova seřazená podle jejich důležitosti.

Funkce NormalizeImportances normalizuje důležitnostní hodnoty do rozpětí 0.5 - 100.

Následuje už jen vypsání slov do standardního či definovaného outputu. Program vypíše prvních 20 klíčových slov dokumentu, a to jednoduchým for loopem.

4 Výsledky

Při práci na projektu jsem kód testoval na několika různých dokumentech. V této kapitole předvedu výsledky na knize Malý princ [2], Wikipedia článku o stíhacích letounech [3] a o dědičnosti [4].

4.1 Malý princ

Celkový počet slov dokumentu: 11 524

Počet odstraněných stop words: 3 691

Počet odstraněných krátkých slov: 1 214

Počet načtených unikátních slov: 3 074

Počet odstraněných neobvyklých unikátních slov: 2 729

Celkový počet analyzovaných unikátních slov: 345

Kličová slova, sestupně podle důležitosti:

princ malý prince beránka planetu malého král baobaby planeta trny beránek sopky pijan princi bydlil slunce svítilnu dodal trochu zasmál

4.2 Stíhací letoun

Celkový počet slov dokumentu: 2 316

Počet odstraněných stop words: 597

Počet odstraněných krátkých slov: 150

Počet načtených unikátních slov: 1 031

Počet odstraněných neobvyklých unikátních slov: 935

Celkový počet analyzovaných unikátních slov: 96

Kličová slova, sestupně podle důležitosti:

stíhací stíhačky letouny záchytné letoun války bombardéry lightning stroje stíhaček fighter např světové suchoj stíhače přepadové generace lockheed výzbrojí studené

4.3 Dědičnost

Celkový počet slov dokumentu: 735

Počet odstraněných stop words: 177

Počet odstraněných krátkých slov: 31

Počet načtených unikátních slov: 392

Počet odstraněných neobvyklých unikátních slov: 366

Celkový počet analyzovaných unikátních slov: 26

Kličová slova, sestupně podle důležitosti:

dědičnost dědičnosti sekvence recesivní heterozygotní dominantní geny rozmnožování alely buňky generace dědičné přenos křížení organismů dělení organismu nazývá informace rozdíly

5 Možnosti zlepšení

V sekci Výsledky je vidět několik nedostatků kódu. Zkratka „např“ se v klíčových slovech akademických článků objevuje relativně často - bylo by možné většinu zkratek z dokumentu plně odstranit při zpracování textu, například odstraněním všech slov, které mají za sebou tečku, po které nenásleduje velké písmeno.

Jedním z velkých rozhodnutí, které jsem při psaní programu musel udělat, byla otázka lematizace. Lematizace je proces převedení jednotlivých tvarů slova do jejich jednotného tvaru. Ve Výsledcích mužeme vidět, že program vrací klíčova slova například „princ prince princi“. Po lematizaci by se všechna taková slova složila do tvaru „princ“. Tento přístup by pravděpodobně zlepšil schopnost programu správně identifikovat klíčová slova; je ale potřeba vzít v potaz, že klíčová slova se podle původního zadání projektu budou dále používat pro hledání podobných dokumentů. Jelikož nelze předpokládat, že texty všech hledaných dokumentů jsou lematizovany, lematizace původního textu by přidala složitý krok.

Nicméně, pokud by se lematizace nakonec implementovala, stačilo by místo korpusu syn2015_word_utf8 použít soubor syn2015_lemma_utf8, který obsahuje lemmata. Pro lematizaci samotnou by se dala využít CPAN knihovna Ufal::MorphoDiTa.

6 Závěr

Program úspěšně extrahuje klíčová slova z českých dokumentů. Funguje nejlépe pro texty velikosti přibližně 2 000 slov. Program jsem testoval na beletrii i na literatuře faktu v podobě Wikipedia článcích.

Zatímco funkčnost kódu by se dala dále vylepšit pomocí poznámek definovaných v sekci Možnosti zlepšení, program i ve svém momentálním stavu exportuje klíčová slova, která jsou pro daný dokument velmi relevantní.

Reference

[1] M. Křen, V. Cvrček, T. Čapka, A. Čermáková, M. Hnátková, D. Kováříková, H. Skoumalová, P. Truneček, P. Vondřička, and A. Zasina, *SYN2015: reprezentativní korpus psané češtiny*. Praha: Ústav Českého národního korpusu FF UK, 2015.

[2] A. de Saint-Exupéry and R. Podaný, *Malý princ*, 14. ed. Praha: Albatros, 1943.

[3] "Stíhací letoun", *Wikipedia: the free encyclopedia*, 2007. [Online]. Dostupné z: https://cs.wikipedia.org/wiki/St%C3%ADhac%C3%AD_letoun.

[4] "Dědičnost", *Wikipedia: the free encyclopedia*, 2017. [Online]. Dostupné z: https://cs.wikipedia.org/wiki/D%C4%9Bdi%C4%8Dnost.