ZADÁNÍ SEMESTRÁLNÍ PRÁCE
JEDNODUCHÝ STEMMER

Zadání

Naprogramujte v ANSI C přenositelnou[1] **konzolovou aplikaci**, která bude pracovat jako tzv. *stemmer*. Stemmer je algoritmus, resp. program, který hledá kořeny slov. Stemmer pracuje ve dvou režimech: (i) v režimu **učení**, kdy je na vstupu velké množství textu (tzv. *korpus*) v jednom konkrétním etnickém jazyce (libovolném) a na výstupu pak slovník (seznam) kořenů slov; nebo (ii) v režimu zpracování slov, kdy je na vstupu slovo (nebo sekvence slov) a stemmer ke každému z nich určí jeho kořen. Tento proces, tzv. *stemming* je jedním ze základních stavebních kamenů nesmírně zajímavého odvětví umělé inteligence, které se označuje jako NLP (= Natural Language Processing, česky zpracování přirozeného jazyka).

Vaším úkolem je tedy implementace takového stemmeru, ovšem velice jednoduchého, podle dále uvedených instrukcí.

Stemmer se bude spouštět příkazem `sistem.exe`[2] ⟨`corpus-file` | [”]`word-sequence`[”]⟩ [`-msl=`⟨*celé číslo*⟩] [`-msf=`⟨*celé číslo*⟩].

Symbol ⟨`corpus-file`⟩ zastupuje jméno vstupního textového souboru s korpusem, tj. velkým množstvím textu, který se použije k „natrénování“ stemmeru. Přípona souboru nemusí být uvedena; pokud uvedena není, předpokládejte, že má soubor příponu `.txt`. Symbol ⟨`word-sequence`⟩ zastupuje slovo nebo sekvenci slov, k nimž má stemmer určit kořeny. Režim činnosti programu je dán předaným parametrem: Je-li parametrem jméno (a případně cesta k) souboru, pak bude stemmer pracovat v režimu učení, tedy tvorby databáze kořenů na základě analýzy dat z korpusu. Je-li parametrem slovo nebo sekvence slov (ta musí být uzavřena v uvozovkách), pak stemmer bude pracovat v režimu zpracování slov, tedy určování kořene každého slova ze sekvence.

Program může být spuštěn se dvěma nepovinnými parametry:

`-msl` – Nepovinný parametr `-msl=`⟨*celé číslo*⟩ specifikuje minimální délku kořene slova (msl = Minimum Stem Length), který bude uložen do databáze kořenů. Není-li tento parametr předán, použije se implicitní minimální délka kořene 3 znaky. Tento parametr je tedy zřejmě použitelný jen v kombinaci s cestou ke korpusu, tedy v režimu učení stemmeru.

`-msf` – Nepovinný parametr `-msf=`⟨*celé číslo*⟩ určuje minimální počet výskytů příslušného kořene (msf = Minimum Stem Frequency). Pokud se tento kořen v korpusu nevyskytl alespoň tolikrát, kolik je určeno tímto parametrem, nepoužije se při zpracování slov, tj. stemmer nemůže u žádného zpracovávaného slova oznámit, že tento kořen je kořenem předmětného slova. Není-li tento parametr předán, použije se implicitní minimální počet výskytů kořene 10×. Tento parametr je tedy zřejmě použitelný jen v kombinaci se slovem nebo sekvencí slov, tedy v režimu zpracování slov.

Program může být během testování spuštěn například takto (režim učení):

```
...\>sistem.exe e:\data\czech-corpus.txt -msl=4
```

[1] Je třeba, aby bylo možné Váš program přeložit a spustit na PC s operačním prostředím Win32/64 (tj. operační systémy Microsoft Windows NT/2000/XP/Vista/7/8/10) a s běžnými distribucemi Linuxu (např. Ubuntu, Mint, OpenSUSE, Debian, atp.). Server, na který budete Vaši práci odevzdávat a který ji otestuje, má nainstalovaný operační systém Debian GNU/Linux 9.1 Stretch s jádrem verze 3.2.78-1 x86_64 a s překladačem gcc 6.3.0.

[2] **sistem** = **si**mple **stem**mer. Přípona `.exe` je povinná i při sestavení v Linuxu, zejm. při automatické kontrole validačním systémem.

A následně v režimu zpracování slov třeba takto:

```
...\>sistem.exe "šel pes do lesa a potkal dlažební kostku" -msf=15
...\>sistem.exe bezdomovec -msf=5
```

Úkolem vašeho programu tedy je v režimu učení vytvořit databázi kořenů (textový soubor, vizte dále) a v režimu zpracování slov vypsat kořen každého slova ze vstupní sekvence. Jak kořeny slov najít bude naznačeno níže.

V případě, že nebude programu předán parametr prvního nebo druhého uvedeného typu, vypište krátké chybové hlášení (anglicky) a oznamte chybu operačnímu prostředí pomocí nenulového návratového kódu[3]. Pokud bude stemmeru při prvním spuštění předáno slovo nebo sekvence slov, ukončete jej chybovým stavem (a krátkým vysvětlujícím hlášením), indikujícím, že nedošlo k předchozímu vytvoření databáze kořenů slov, a tudíž není možné u slov ze sekvence jejich kořeny určit.

Hotovou práci odevzdejte v jediném archivu typu ZIP prostřednictvím automatického odevzdávacího a validačního systému. Archiv nechť obsahuje všechny zdrojové soubory potřebné k přeložení programu, **makefile** pro Windows i Linux (pro překlad v Linuxu připravte soubor pojmenovaný `makefile` a pro Windows `makefile.win`) a dokumentaci ve formátu PDF vytvořenou v typografickém systému TEX, resp. LATEX. Bude-li některá z částí chybět, kontrolní skript Vaši práci odmítne.

Formáty vstupních a výstupních souborů a vstupu z konzole

Program bude pracovat **výhradně s 1-bytovým (8mi-bitovým) kódováním znaků**: Neuvažujte žádné kódování znaků národních abeced, které používá více než jeden byte na znak nebo proměnný počet bytů, tj. žádná kódování z rodiny Unicode (UTF-8, UTF-16, UCS-2, apod.).

Vzhledem k tomu, že ukázkový korpus na webu je v češtině, připadá v úvahu prakticky pouze kódování Windows/CP-1250 nebo ISO-8859-2 (ISO Latin 2). Stemmeru by to mělo být úplně jedno, pokud dodržíte pravidlo „jeden byte kóduje jeden znak“. Bude-li korpus užitý pro natrénování složen např. z německých textů, tj. znaky budou kódovány podle Windows/CP-1252, pak i výstupní soubor s databází kořenů bude kódován CP-1252. V režimu zpracování slov pak bude stemmer určovat kořeny jen u slov předaných v tomto kódování. Bude-li na vstupu slovo obsahující znaky národních abeced v jiném kódování, než ve kterém je databáze kořenů, pak pochopitelně kořen nebude (protože nemůže být) nalezen. Jednoduše řečeno, programátor do kódování znaků nemusí vůbec zasahovat...

Stejně tak při vstupu slova nebo sekvence slov z konzole můžete předpokládát, že uživatel má konzoli správně nastavenou na příslušné kódování znaků národních abeced tak, aby bylo shodné s kódem použitým v korpusu (a tedy i v databázi kořenů). Při testování programu bude použit pouze anglický korpus (tj. 7-bit ASCII) a český v kodování Win-1250.

Postup tvorby databáze kořenů slov

Vámi implementovaný stemmer nebude pravidlový (jako je např. Porter Stemmer, uvedený na konci v odstavci Užitečné techniky a odkazy), ale statistický. To znamená, že kořeny slov odvodí analýzou velkého množství textu v jednom konkrétním etnickém jazyce (ukázkový korpus přiložený na webu k tomuto zadání je v češtině a jde o texty Karla Čapka).

Z textu nechť váš program nejprve vytvoří frekvenční slovník, tzn. každé slovo, které se v textu vyskytuje, bude v tomto slovníku uvedeno právě jednou a bude k němu přiřazen počet výskytů v textu. Vzhledem k tomu, že tento slovník bude váš algoritmus v dalších krocích intenzivně prohledávat, zvolte pro jeho implementaci vhodnou vnitřní reprezentaci (např. spojový seznam je **hodně špatný** nápad).

[3] Stejně tak čiňte i v případě jiných chyb, např. nesprávném formátu vstupního souboru a podobně.

Jakmile je frekvenční slovník připravený, budete hledat kořeny slov, a to tak, že každé slovo porovnáte se všemi ostatními slovy ve slovníku a zjistíte, jaké mají nejdelší společné podřetězce. Společné podřetězce kratší než je zvolená minimální délka kořene, určená nepovinným parametrem `-msl=`⟨*celé číslo*⟩, ignorujte. Společné podřetězce, které délkou vyhovují, vkládejte do databáze – frekvenčního slovníku kořenů (může být implementován stejnou technikou jako frekvenční slovník slov v korpusu). Pokud už kořen v databázi existuje, nevkládejte jej znovu, ale zvyšte příslušný čítač počtu výskytů tohoto kořene.

Vnitřní reprezentace databáze kořenů není předepsána, ale není pochopitelně možné využívat služby externího databázového serveru nebo knihovny – vše musí být implementováno ve vašem programu, aby nebyl závislý na určité specifické konfiguraci systému, kde poběží. S ohledem na výše uvedené je ale zřejmé, že např. prosté pole řetězců nebo spojový seznam nejsou vhodné implementace z důvodů algoritmické složitosti prohledávání takových struktur.

Databázi kořenů, vytvořenou popsaným způsobem, váš program před dokončením běhu v režimu učení **uloží na disk** do textového souboru s názvem `stems.dat`, organizovaného takto:

```
andel 5
angl 28
anick 7
aparat 3
⋮
bav 35
beh 19
⋮
zac 42
zada 18
zadn 53
⋮
```

Kořeny budou seřazeny podle ASCII hodnot jednotlivých znaků v každém řetězci, **vzestupně**. Při řazení uvažujte výhradně ASCII hodnotu daného znaku, bez ohledu na to, kam je případně zařazen v národní abecedě. Na každém řádku bude právě jeden kořen v kódování shodném s kódováním vstupního souboru s korpusem, následuje jedna mezera a celé dekadické číslo, vyjadřující počet výskytů daného kořene, tj. vlastně počet vložení tohoto kořene do databáze během procesu učení.

Postup zpracování slov

Vstupní sekvenci slov rozdělíte na jednotlivá slova. Pak pro každé slovo budete procházet databázi kořenů a budete hledat nejdelší kořen, který se celý nachází ve zkoumaném slově jako jeho podřetězec. Tento kořen pak bude určen jako kořen zkoumaného slova.

Pokud ovšem určený kořen nebyl pozorován při tvorbě databáze alespoň tolikrát, kolik je určeno nepovinným parametrem `-msf=`⟨*celé číslo*⟩ (nebo jeho implicitní hodnotou), není možné ho vybrat jako výsledný kořen daného slova. V takovém případě hledejte další nejdelší kořen s dostatečným počtem výskytů. Pokud ani toto nepovede k nalezení použitelného kořene, oznámí stemmer v předepsaném formátu (vizte dále), že kořen takového slova nelze určit.

Formát výstupu na konzoli v režimu zpracování slov

```
...\>sistem.exe "šel pes do lesa a potkal dlažební kostku" -msf=15
šel -> šel
pes -> pes
```

```
do␣->␣0
lesa␣->␣les
a␣->␣0
potkal␣->␣potk
dlažební␣->␣dlaž
kostku␣->␣kostk
```

Užitečné techniky a odkazy

Uvedené techniky je možné (ale nikoliv nezbytně nutné) využít při řešení úlohy. Protože se jedná o postupy víceméně standardní, lze k nim nalézt velké množství dokumentace:

1. Datová struktura Trie (česky prefixový strom) – popis techniky na Wikipedii, `https://cs.wikipedia.org/wiki/Trie`
2. Techniky implementace množin
3. Algoritmus LCS (Longest Common Substring)
4. Stemming – popis techniky na Wikipedii, `https://en.wikipedia.org/wiki/Stemming`
5. Porter Stemmer, Lovins Stemmer, czech_stemmer

Řešení úlohy je zcela ve vaší kompetenci – zvolte takové algoritmy a techniky, které podle vás nejlépe povedou k cíli.

Poznámka: Tento stemmer je velice jednoduchý, doslova triviální. Nečekejte proto, že bude hledat kořeny slov správně, tak jak to známe z hodin českého jazyka na základní či střední škole. Důležité je, aby je hledal v naprosté shodě s výše popsaným postupem.