FAKULTA APLIKOVANÝCH VĚD
ZÁPADOČESKÉ UNIVERZITY
V PLZNI

Identifikace spamu naivním bayesovským klasifikátorem

Semestrální práce KIV/PC

Štěpán Faragula
A21B0119P

31. prosince 2022

Obsah

Kapitola 1

Zadání

Při volbě zadání semestrální práce jsme měli na výběr z následujících možností:

1. Hledání kořenů rovnice
2. Identifikace spamu naivním bayesovským klasifikátorem
3. Celočíselná kalkulačka s neomezenou přesností

V této práci je popsáno řešení práce **číslo 2**.

1.1 Detaily zadání

Naprogramujte v ANSI C přenositelnou **konzolovou aplikaci**, která bude **rozhodovat, zda úsek textu** (textový soubor předaný jako parametr na příkazové řádce) **je nebo není spam**.

Program bude přijímat z příkazové řádky celkem **sedm** parametrů: První dva parametry budou vzor jména a počet trénovacích souborů obsahujících nevyžádané zprávy (tzv. **spam**). Třetí a čtvrtý parametr budou vzor jména a počet trénovacích souborů obsahujících vyžádané zprávy (tzv. **ham**). Pátý a šestý parametr budou vzor jména a počet testovacích souborů. Sedmý para- metr představuje jméno výstupního textového souboru, který bude po dokončení činnosti Vašeho programu obsahovat výsledky klasifikace testovacích souborů.

Program se tedy bude spouštět příkazem

`spamid.exe` ⟨`spam`⟩ ⟨`spam-cnt`⟩ ⟨`ham`⟩ ⟨`ham-cnt`⟩ ⟨`test`⟩ ⟨`test-cnt`⟩ ⟨`out-file`⟩ ↵

Symboly ⟨`spam`⟩, ⟨`ham`⟩ a ⟨`test`⟩ představují vzory jména vstupních souborů. Symboly ⟨`spam-cnt`⟩, ⟨`ham-cnt`⟩ a ⟨`test-cnt`⟩ představují počty vstupních souborů. Vstupní soubory mají následující pojmenování: `vzorN`, kde `N` je celé číslo z intervalu $\langle 1;N \rangle$.

Přípona všech vstupních souborů je `.txt`, přípona není součástí vzoru. Váš program tedy může být během testování spuštěn například takto:

```
spamid.exe spam 10 ham 20 test 50 result.txt ⏎
```

Výsledkem činnosti programu bude textový soubor, který bude obsahovat seznam testovaných souborů a jejich klasifikaci (tedy rozhodnutí, zda je o spam či neškodný obsah – ham).

Pokud nebude na příkazové řádce uvedeno právě sedm argumentů, vypište chybové hlášení a stručný návod k použití programu v angličtině podle běžných zvyklostí (viz např. ukázková semestrální práce na webu předmětu Programování v jazyce C). **Vstupem programu jsou pouze argumenty na příkazové řádce – interakce s uživatelem pomocí klávesnice či myši v průběhu práce programu se neočekává.**

Hotovou práci odevzdejte v jediném archivu typu ZIP prostřednictvím automatického odevzdávacího a validačního systému. Postupujte podle instrukcí uvedených na webu předmětu. 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.

Kapitola 2

Analýza úlohy

V úloze máme za úkol **zařadit soubory** do jedné ze dvou tříd – **spam** či **ham**. Je nám výrazně doporučeno použít **naivní bayesovský klasifikátor**, není tedy důvod rozebírat, jaký způsob klasifikace zvolíme.

2.1 Naivní bayesovský klasifikátor

Algoritmus, který má dvě fáze – **fáze učení** a **fáze klasifikace**.

2.1.1 Fáze učení

V této fázi vycházíme z předpokladu, že máme k dispozici **trénovací soubory** obsahující pouze slova označena jako spam nebo ham. Každý soubor přečteme a vytvoříme tak **slovník klasifikátoru**. U každého slova budeme uchovávat informaci o jeho **počtu výskytu** v trénovacích souborech a jeho **podmíněnou pravděpodobnost výskytu**. Takto vytvořený slovník by měl ještě obsahovat **apriorní pravděpodobnost**, která analýzou vstupních souborů stanoví **výchozí pravděpodobnost výskytu spamu či hamu**.

2.1.2 Fáze klasifikace

Testovací soubory budeme klasifikovat podle obsažených slov. Průběh klasifikace je takový, že každému slovu vyskytující se v **testovacím souboru a zároveň ve slovníku klasifikátoru** přiřadíme **podmíněnou pravděpodobnosti výskytu** slova ve spamu a hamu. Dále sečteme logaritmy všech **podmíněných pravděpodobností** slov a přičteme logaritmus **apriorní pravděpodobnosti** třídy. Soubor zařadíme do té třídy, která bude mít **největší výsledek**. Klasifikace souboru je popsána rovnicí 2.1

$$c = \arg\max_{c_i \in C} \left(\log(P(c_i)) + \sum_{k \in \textbf{pozice}} \log(P(\langle \text{word}_k | c_i \rangle)) \right) \tag{2.1}$$

kde:

- C - množina všech tříd
- c_i - označení třídy (spam, ham)
- $P(c_i)$ - apriorní pravděpodobnost dané třídy
- $P(\langle \text{word}_k | c_i \rangle)$ - podmíněná pravděpodobnost výskytu slova

2.2 Definice problému

Klasifikátor potřebuje ke své činnosti **slovník**. **Slovník** můžeme vnímat jako **datovou strukturu**, která dokáže vrátit výsledek podle hledaného klíče (slova). Chceme tedy, aby tato struktura vkládala a vyhledávala slova **co nejrychleji** pro obsáhlý slovník. Jelikož úlohu vytváříme v ANSI C, musíme si tuto strukturu naprogramovat sami.

Dále potřebujeme vyřešit **způsob klasifikace souborů**.

2.3 Volba datové struktury pro slovník

Slovník bude obsahovat informaci o všech **unikátních slov** ze souboru. Ve slovníku nepotřebujeme mazat položky, stačí nám pouze funkce **přidání** a **hledání**. Chceme tedy, aby tyto operace pracovali co nejrychleji.

2.3.1 Dynamické pole

2.3.2 Hash tabulka

2.3.3 Trie

2.4 Způsob klasifikace

Zde se nabízí dvě možnosti: (i) Vytvoříme ještě jeden **slovník** obsahující slova testovacího souboru, který následně **porovnáme se slovníkem klasifikátoru**, (ii) nebo jednotlivá slova budeme porovnávat tzv. *on the fly* způsobem, kde **postupným čtením slov** rovnou klasifikujeme soubor jako spam či ham **bez vytváření dalšího slovníku**. Výhoda přístupu (i) je uchování čitelnosti programu za cenu vyšší paměťové náročnosti. Výhoda přístupu (ii) je jednoduchost implementace bez nároků na paměť.

Kapitola 3

Popis implementace

Příliš žluťoučký kůň úpěl ďábelské ódy.

Kapitola 4

Uživatelská příručka

Příliš žluťoučký kůň úpěl ďábelské ódy.

Kapitola 5

Závěr

Příliš žluťoučký kůň úpěl ďábelské ódy.