

# ZADÁNÍ SEMESTRÁLNÍ PRÁCE

## IDENTIFIKACE SPAMU

### NAIVNÍM BAYESOVSKÝM KLASIFIKÁTOREM

---

## Zadání

Naprogramujte v ANSI C přenositelnou<sup>1</sup> **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 parametr představuje jméno výstupního textového souboru, který bude po dokončení činnosti Vašeho programu obsahovat výsledky klasifikace testovacích souborů.

Program se tedy bude spouštět příkazem

```
spamid.exe2 <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, vypíše 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 programem se neočekává.**

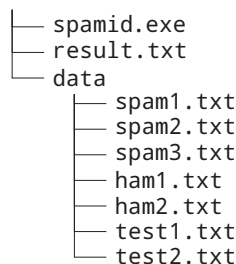
Hotovou práci odevzdejte v jediném archivu typu ZIP prostřednictvím automatického odevzdávacího a validačního systému. Postupujte podle instrukcí uvedených na webu předmětu. Archiv nechtě obsahuje všechny zdrojové soubory potřebné k přeložení programu, **makefile** pro Windows i Linux (pro překlad v Linuxu připravte soubor pojmenovaný **makefile** a pro Windows **makefile.win**) a dokumentaci ve formátu PDF vytvořenou v typografickém systému **TeX**, resp. **L<sup>A</sup>TeX**. Bude-li některá z částí chybět, kontrolní skript Vaši práci odmítne.

## Specifikace vstupu programu

Vstupem programu jsou **pouze parametry na příkazové řádce** – výše zmíněných 7 povinných parametrů, které specifikují trénovací množiny spamu a hamu, testovací množinu a výstupní sou-

<sup>1</sup>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/7/8/10/11) a s běžnými distribucemi Linuxu (např. Ubuntu, Debian, Red Hat, atp.). Server, na který budete Vaši práci odevzdávat a který ji otestuje, má nainstalovaný operační systém Debian GNU/Linux 10 (buster) s jádrem verze 4.19.0-11-amd64 a s překladačem gcc 8.3.0.

<sup>2</sup>Přípona **.exe** je povinná i při sestavení v Linuxu, zejm. při automatické kontrole validačním systémem.



Obrázek 1: Ukázka adresářové struktury

bor s klasifikací textu – **interakce** s uživatelem **pomocí klávesnice či myši v průběhu práce programem se neočekává.**

Každý vstupní soubor obsahuje jeden řádek slov oddělených znakem mezera (ASCII #32/0x20). Vstupní soubory jsou již připravené pro klasifikaci, není tedy nutné provádět žádné další předzpracování. Vstupní soubory obsahují pouze písmena anglické abecedy v kódování ASCII (7-bit, tj. není nutné řešit kódování znaků národních abeced).

Parametry programu na příkazové řádce `<spam>`, `<ham>` a `<test>` nejsou přímo jména vstupních textových souborů, ale vzory, podle kterých se tato jména tvoří. Je-li tedy např. parametr `<spam>` zadán ve tvaru `'spam'`, znamená to, že program bude načítat všechny soubory, jejichž jméno je tvořeno řetězcem `'spam'`, za nímž následuje dekadické celé číslo (nezarovnané, nedoplněné nulami) s příponou `'.txt'`. Tomuto vzoru tedy odpovídají např. jména souborů `'spam1.txt'`, `'spam2.txt'`, ..., `'spam10.txt'`, `'spam11.txt'`, ..., `'spam100.txt'`, `'spam101.txt'`, atd. Vstupní soubory budou umístěny v podadresáři `data`, který bude ve stejném adresáři jako spustitelný soubor. Adresářová struktura je naznačena na obrázku 1.

## Specifikace výstupu programu

Úkolem programu je u všech specifikovaných testovacích souborů určit, **zda se jedná o nevyžádanou zprávu (spam) nebo ne**. Systémy tohoto typu využívají např. poskytovatelé služeb elektronické pošty pro automatické třídění příchozí pošty. Jedná se o jednoduchou úlohu z oblasti strojového učení, konkrétně o učení s učitelem. To znamená, že se algoritmus na základě předložených (korektně označených) trénovacích dat naučí rozpoznávat data, která do té doby neviděl. Pro tuto konkrétní úlohu lze úspěšně využít tzv. *naivní bayesovský klasifikátor* (NBK), který bude detailně popsán dále v sekci .



Výstup programu bude směřován do textového souboru, daného posledním parametrem programu na příkazové řádce. Tento soubor bude obsahovat na každém řádku jméno testovaného souboru a jeho klasifikaci. Pokud je soubor klasifikován jako spam, bude označen písmenem `'S'` (spam), v případě, že se nejedná o spam, bude označen písmenem `'H'` (ham), a to v následujícím formátu: `<jméno souboru>`, `<tabulátor>`, `<klasifikace>` a `<znak konce řádku>`.

Ukázka níže demonstuje, jak může vypadat jeden konkrétní výstupní soubor:

```

test1.txt→H\r\n3
test2.txt→H\r\n
test3.txt→S\r\n
...
  
```

<sup>3</sup>Sekvence znaků `\r\n` bude použita pochopitelně jen v případě OS Windows. Platformně nezávislého výsledku dosáhnete využitím escape sekvence `'\n'`.

test110.txt → S   
test111.txt → H 

Na začátku ani na konci souboru nenechávejte žádné bílé znaky (mezery, tabulátory, apod.) či prázdné řádky. Každá řádka začíná jménem souboru a končí znakem konce řádku.

Klasifikační algoritmy pochopitelně nepracují se 100%-ní spolehlivostí. Při zpracování výsledků Vaší semestrální práce je proto **minimální hranice přesnosti klasifikace stanovena na 90 %**. Jinými slovy, ze 100 testovacích dokumentů musí Váš program správně klasifikovat nejméně 90 dokumentů, aby byla semestrální práce přijata jako vyhovující. Při použití *naivního bayesovského klasifikátoru* není problém této přesnosti dosáhnout – ovšem Vaší kreativitě se nekladou žádná omezení, takže pokud algoritmus NBK využít nechcete, můžete samozřejmě využít jiný, nebo navrhnout vlastní.

## Algoritmus naivního bayesovského klasifikátoru

Při použití *naivního bayesovského klasifikátoru* (NBK) můžete pro potřeby semestrální práce implementovat algoritmus popsáný níže. Algoritmus má dvě fáze: (i) Fázi učení, kdy analýzou vstupní trénovací množiny správně označeného textu (tj. ke každému vstupnímu textovému souboru existuje správná informace o tom, zda se jedná o spam či ham) vytvoří “slovník” podmíněných pravděpodobností výskytu slova v textu za podmínky klasifikace daného textu do konkrétní třídy. Ve (ii) fázi klasifikace pak neznámý soubor klasifikuje tak, že vypočítá kumulativní<sup>4</sup> podmíněné pravděpodobnosti výskytu slov daného souboru za podmínky jeho klasifikace do třídy *spam* a za podmínky jeho klasifikace do třídy *ham*. Ta, z těchto dvou podmíněných pravděpodobností, která je vyšší, vyhrává, a soubor je zařazen do vítězné třídy.

Obě dvě popsané fáze jsou níže zapsány v pseudokódu. Fázi učení popisuje pseudokód Alg 1, fázi klasifikace popisuje pseudokód Alg 2.

### Učení

NBK vypočítává pravděpodobnost příslušnosti dokumentu ke třídě, pro vypočtení této pravděpodobnosti je nutné nejprve vypočítat následující hodnoty, pro každou klasifikační třídu:

- $P(c)$  – apriorní pravděpodobnost výskytu klasifikační třídy  $c$  v datech. Pokud by data byla hody mincí, tak pravděpodobnost třídy *hlava* je 0,5, pravděpodobnost třídy *orel* je také 0,5 a pravděpodobnost třídy *je\_jasno* je 0, protože taková třída se v datech neobjevuje. V naší úloze jsou 2 třídy: *spam* a *ham* – je tedy třeba analýzou vstupní trénovací množiny stanovit, jaká je pravděpodobnost jejich výskytu, a to z jejich relativní četnosti v trénovací množině.
- $P(\langle \text{word} \rangle | c)$  – pravděpodobnost, že se slovo  $\langle \text{word} \rangle$  vyskytne v dokumentu za podmínky klasifikace do třídy  $c$ . V tomto příkladu využíváme tzv. *bag-of-words model*, kdy neuvažujeme pozici slova v dokumentu, stačí, že se v dokumentu vyskytuje. Je tedy zřejmé, že ke každému unikátnímu slovu z celé trénovací množiny budou po provedení tohoto kroku existovat 2 vypočítané hodnoty, a to:  $P(\langle \text{word} \rangle | c = \text{'spam'})$  a  $P(\langle \text{word} \rangle | c = \text{'ham'})$ .

### Klasifikace

Následující algoritmus navazuje na algoritmus Alg 1, tzn. **slovník** je získaný na řádce 4 a  $P(\langle \text{word}_k \rangle | c_i)$  je podmíněná pravděpodobnost vypočtená na řádce 13.

<sup>4</sup>Tedy souhrnné, za celý dokument, kdy každé slovo dokumentu přispěje svojí vlastní pravděpodobností do součinu celkové podmíněné pravděpodobnosti.

---

**Alg 1** Učení naivního bayesovského klasifikátoru

---

```
1: ▷ Množina C obsahuje dvě podmnožiny  $\{c_{spam}, c_{ham}\}$ 
2: def NBLEARNTEXT(trénovací množina, C)
3:   ▷ Vytvoření slovníku
4:   slovník[slovo, četnost]  $\leftarrow$  unikátní slova a jejich počet z trénovací množiny
5:   ▷ Výpočet pravděpodobností
6:   for  $\forall$  klasifikace  $c_i \in C$  :
7:     doci  $\leftarrow$  dokumenty z trénovací množiny, patřící do třídy  $c_i$ 
8:      $P(c_i) = |\mathbf{doc}_i|/|\mathbf{trénovací množina}|$ 
9:     texti  $\leftarrow$  sloučení všech doci do jediného dokumentu
10:     $n \leftarrow$  počet slov v texti (k-duplicity se započítávají k-krát)
11:    for  $\forall$  slova  $\langle \text{word}_k \rangle$ ,  $k \in \langle 1, |\mathbf{slovník}| \rangle$  ve slovník :
12:       $n_k \leftarrow$  počet výskytů slova  $\langle \text{word}_k \rangle$  v texti
13:       $P(\langle \text{word}_k \rangle | c_i) \leftarrow (n_k + 1)/(n + |\mathbf{slovník}|)$ 
```

---

---

**Alg 2** Klasifikace neznámého vzorku

---

```
1: def NBCLASSIFYTEXT(doc, slovník)
2:   pozice  $\leftarrow$  všechny pozice slov v doc obsažených ve slovník
3:   return  $c_{NB}$ 
```

---

$$c_{NB} = \underset{c_i \in C}{\operatorname{argmax}} \left( P(c_i) \times \sum_{k \in \mathbf{pozice}} \log(P(\langle \text{word}_k \rangle | c_i)) \right) \quad (1)$$

Rovnice (1) popisuje, jak získat klasifikaci neznámého vzorku. Funkce  $\operatorname{argmax}$  funguje tak, že pro všechny hodnoty  $c_i$  vyčíslí hodnotu argumentu a vrátí  $c_i$ , pro kterou byla vypočtená hodnota nejvyšší. Při vyhodnocování argumentu funkce  $\operatorname{argmax}$  věnujte pozornost uvedené sumaci. V literatuře se spíše setkáte s následujícím zápisem:

$$P(c_i) \times \prod_{k \in \mathbf{pozice}} P(\langle \text{word}_k \rangle | c_i),$$

kde symbol  $\prod$  označuje násobení namísto sčítání, jako v případě sumace. Jelikož jsou podmíněné pravděpodobnosti jednotlivých slov velmi malá čísla ( $10^{-1}$  až  $10^{-7}$  a menší), došlo by při násobení těchto čísel ke ztrátě přesnosti vzhledem ke způsobu uložení reálných čísel v počítači. Pokud však pravděpodobnosti zlogaritmuje, zamezíme tomuto problému, aniž by algoritmus ztratil na funkčnosti.

## 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. Bayesovské učení,
2. naivní bayesovský klasifikátor,
3. hash tabulka (tabulka s rozptýlenými položkami),
4. trie (prefixový strom),
5. binární vyhledávací strom.

**Řešení úlohy je zcela ve vaší kompetenci** – zvolte takové algoritmy a techniky, které podle vás nejlépe povedou k cíli. Upozorňuji, že není dobrý nápad uchovávat slovník unikátních slov a jejich četností ve spojovém seznamu – program by v takovém případě pracoval velmi pomalu.