Západočeská univerzita v Plzni
Fakulta aplikovaných věd
Katedra informatiky a výpočetní techniky

KIV/SU

Spam filtr

Plzeň 2016 Jan Schröpfer

1 Úvod

Tato semestrální práce se zabývá problémem filtrování spamu pomocí naivního Bayesovského klasifikátoru. Jedná se tedy o problém rozdělení textu na samostatná slova, určení pravděpodobnosti, zda se tyto slova vyskytují častěji ve spamu či v neškodných e-mailech a podle těchto určených pravděpodobností zjišťovat, s jakou pravděpodobností je zadaný text e-mailu škodlivý.

Pro implementaci spam filtru byl zvolen jazyk C# a vývojové prostředí Visual Studio 2010, jako algoritmus byl zvolen algoritmus Paula Grahama pro naivní Bayesovský klasifikátor, pro vytvoření dokumentace byl použit jazyk TeX a editor TeXmaker.

2 Teoretický rozbor

Pro zadaný problém (filtrování spamu) jsem se rozhodl použít přístup Paula Grahama a jeho spam filtru. Ten používá naivní Bayesovský klasifikátor pro určení pravděpodobnosti, že text je spam.

2.1 Důležité pojmy

Korpus - Z latinského slova *corpus*, jedná se o velké množství textu, které má obdobný význam. V našem případě se jedná o velké množství e-mailových textů.

Token - Token je v našem případě jedno konkrétní slovo textu. Každý jednotlivý token je case sensitive (z důvodu, že v podvodných e-mailech se často vyskytují například běžná slova celá psaná velkými písmeny)

SPAM - Jedná se o masově šířený text internetem, který je nevyžádaný. Původně se jednalo pouze o reklamy, později se spamem začali šířit viry a podvody.

HAM - Jedná se o text, který není závadný (opak spamu). To může znamenat například e-maily od rodiny, banky, učitele apod..

2.2 Výpočet pravděpodobnosti slova

Na začátku je třeba mít dvě učící množiny (korpusy) - jednu obsahující dobré e-maily (ham) a druhou obsahující spam. Poté se vytvoří dvě tabulky obsahující tokeny (jednotlivá slova) a počet jejich výskytů v ham a spam korpusu (prozatím individuálně). Nakonec je třeba vytvořit třetí tabulku, která obsahuje pravděpodobnosti jednotlivých tokenů. Pravděpodobnost, že token patří do spamu se podle Grahama vypočítá algoritmem:

```
(let ((g (* 2 (or (gethash word good) 0)))
        (b (or (gethash word bad) 0)))
    (unless (< (+ g b) 5)
        (max .01 (min .99 (float (/ (min 1 (/ b nbad))
                                    (+ (min 1 (/ g ngood))
                                    (min 1 (/ b nbad)))))))))
```

kde *word* je token, jehož pravděpodobnost počítáme, *good* a *bad* jsou tabulky spamu a hamu a *ngood* a *nbad* jsou počty výskytů v hamu a spamu.

Algoritmus tedy dělá následující:

- Počet výskytů v hamu pro každý token zdvojnásobí. To se dělá z toho důvodu, že slova která se vyskytují častěji v hamu než ve spamu se většinou tak často neopakují v e-mailech.
- Pravděpodobnost se počítá pouze pro slova, která se vyskytují v učící množině minimálně 5-krát (díky zdvojnásobení v hamu alespoň 3-krát).
- Slova, která se vyskytují pouze v hamu mají pravděpodobnost 0.01.
- Slova, která se vyskytují pouze ve spamu mají pravděpodobnost 0.99.

2.3 Výpočet pravděpodobnosti textu

Pro výpočet pravděpodobnosti, že text je opravdu spam, se nepoužijí všechna slova e-mailu, ale 20 "nejzajímavějších"slov. Za zajímavá slova se považují ta slova, která jsou nejvíce vzdálená od pravděpodobnosti 0.5 (Tedy jejich pravděpodobnost je nejblíže k 0 nebo 1). Slovům, které nejsou ani v ham či spam korpusu, se přiřadí pravděpodobnost 0.4. Pro tyto slova se následně použije vzorec naivního Bayesovského klasifikátoru ve tvaru:

$$\frac{p1p2...p20}{p1p2...p15+(1-p1)(1-p2)...(1-p20}$$

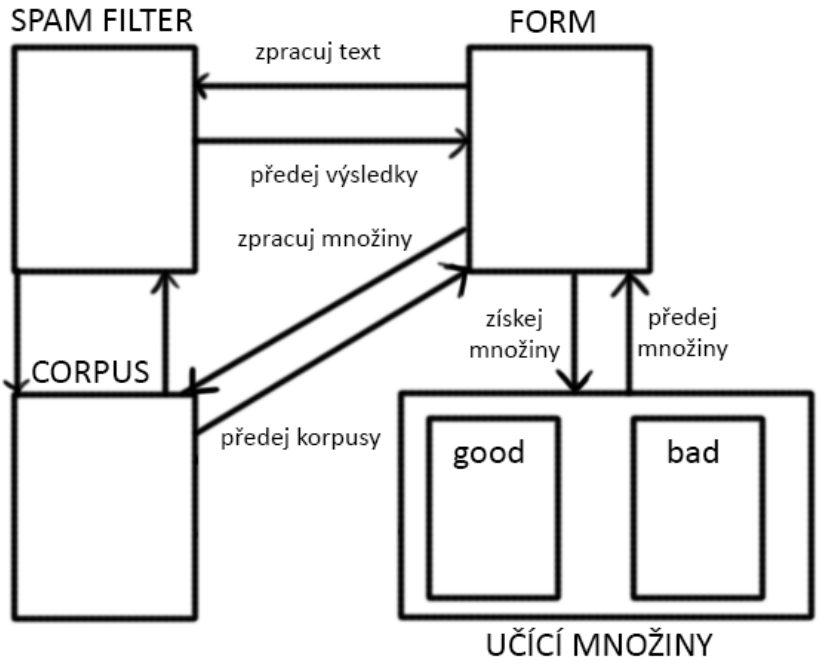
kde *p1, p2. . . p20* jsou pravděpodobnosti spamu 20 nejzajímavějších slov. K 20 slovům došel ve svém článku Paul Graham metodou pokus omyl. Podle jeho testů stačí dokonce i 15 slov z celého e-mailu k tomu, aby program dosáhl přesnosti 99,5% (při dostatečně velkých korpusech spamu a hamu).

3 Implementace

Řešení bylo vytvořeno v jazyce C#. Celkem lze rozdělit řešení do 4 částí (viz obr.3.1):

- Corpus - Třída zabývající se korpusem. obsahuje funkce pro načtení textu a rozdělení na tokeny.
- SpamFilter - Třída obsahující konstanty pro Bayesův algoritmus, funkce pro výpočet pravděpodobnosti spamu a načítání/zapisování dat do souborů
- good.txt, spam.txt - korpusy obsahující spam a ham. Jde tedy o učící množiny spam filtru. obě množiny obsahují několik stovek e-mailů a dají se dále zvětšovat pro navýšení přesnosti výpočtu pravděpodobnosti.
- form - Třída obsahující GUI (grafické uživatelské rozhraní) a volá metody pro načtení učících množin, předává je ke zpracování na korpusy a nakonec slouží k načtení uživatelských textů a jejich testování na spam.

Obrázek 3.1: Graf tříd



3.1 Corpus

Corpus.cs je třída, která vytváří korpusy s tokeny - slovy a jejich četnostmi. Jako taková obsahuje funkce pro zpracování textu z text readeru, vytvoření setříděného slovníku tokenů a přidánim jejich četnosti v daném korpusu.

3.2 Spam filter

Třída ***SpamFilter.cs*** je nejdůležitější třídou celého řešení. Obsahuje metody pro zpracování testovacích množin, výpočet pravděpodobností tokenů a nakonec výpočet pravděpodobnosti toho, zda je text spam. v této sekci si dále popíšeme nejdůležitější části třídy ***SpamFilter.cs***

3.2.1 ListAtributu

ListAtributu je vnitřní třídou třídy ***SpamFilter.cs*** obsahující důležité atributy pro spam filter podle Grahama. Mezi tyto atributy patří:

- VahaDobrehoTokenu - násobek pro dobrý token, defaultně 2.
- MinTokenuProPridani - minimální počet výskytů tokenu ve spamu a hamu, aby se spočítala jeho pravděpodobnost, defaultně 5.
- MinSkore - minimální pravděpodobnost spamu, kterou může token mít (z toho důvodu, že i slova, která se běžně vyskytují jen v hamu můžou být součástí spamu), defaultně 0,011.
- MaxSkore - maximální pravděpodobnost spamu, kterou může token mít. Defaultně 0,99.
- PocetProJistySpam - počet výskytů tokenu ve spamu, aby se prohlásil za vždy spamové slovo, defaultně 10.
- PocetZajimavychSlov - počet zajímavých tokenů, které v testovaném textu zkoumáme. Defaultně 20 (podle článku Paula Grahama lze použít i 15 slov pro dosažení vysoké přesnosti).

3.2.2 SpoctiPravdepodobostTokenu

Jedná se o funkci implementující Grahamův algoritmus (obr. 3.2). Algoritmus obsahuje následující proměnné:

- d - počet výskytů tokenu v hamu

- s - počet výskytů tokenu ve spamu
- dobryfaktor - faktor spočítaný vzorcem $min(1, d/pocetHamTokenu)$
- spatnyfaktor - faktor spočítaný vzorcem $min(1, s/pocetSpamTokenu)$

z těchto proměnných se následně vypočítá pravděpodobnost tokenu přes vzorec:

$$max(MinSkore, min(MaxSkore, \frac{spatnyfaktor}{dobryfaktor+spatnyfaktor}))$$

pokud se token vyskytuje pouze ve spamu (hodnota d je 0) tak se nastaví jeho pravděpodobnost na 0.9998 nebo 0.9999 pokud se vyskytuje ve spamu alespoň 10-krát.

```
int d = _dobry.Tokeny.ContainsKey(token) ? _dobry.Tokeny[token] * listAtributu.VahaDobrehoTokenu : 0;
int s = _spatny.Tokeny.ContainsKey(token) ? _spatny.Tokeny[token] : 0;

if (d + s >= listAtributu.MinTokenuProPridani)
{
    double dobryfaktor = Min(1, (double)d / (double)_pocetDobry);
    double spatnyfaktor = Min(1, (double)s / (double)_pocetSpatny);

    double pravdepodobnost = Max(listAtributu.MinSkore,
                 Min(listAtributu.MaxSkore, spatnyfaktor / (dobryfaktor + spatnyfaktor))
             );

    // Specialni pripad kdyz se token vyskytuje jen ve spamu
    // .9998 pro tokeny jen ve spamu .9999 pokud se objevil vic jak 10krat
    if (d == 0)
    {
        pravdepodobnost = (s > listAtributu.PocetProJistySpam) ? listAtributu.JistySpam : listAtributu.PravdepodobnySpam;
    }

    _pravdepodobnosti[token] = pravdepodobnost;
}
```

Obrázek 3.2: Grahamův algoritmus

3.2.3 Test

funkce Test zjišťuje pravděpodobnost toho, zda je spam či ne podle 20 ”nejzajímavějších”slov - 20 tokenů, které mají pravděpodobnost nejblíže k 0 či 1. Funkce implementuje naivní Bayesovský klasifikátor (obr 3.3) a obsahuje proměnné:

- nasob - součin všech pravděpodobností, že token je spam
- komb - součin všech pravděpodobností, že token není spam
- pravdepodobnosti - pole pravděpodobností nejzajímavějších slov textu

pomocí těchto proměnných se nakonec vypočítá pravděpodobnost, že text je spam přes vzorec $\frac{nasob}{nasob+komb}$

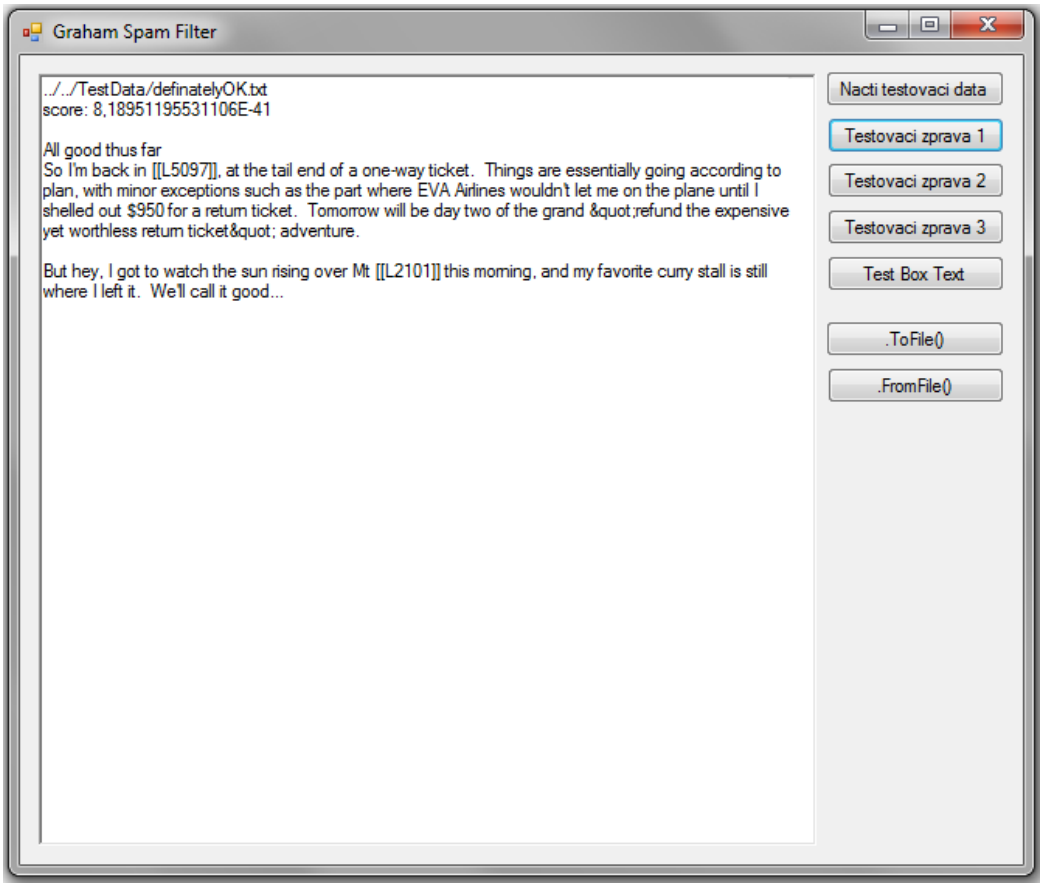
```
double nasob = 1;  // abc..n
double komb = 1;  // (1 - a)(1 - b)(1 - c)..(1-n)
index = 0;
foreach (string key in pravdepodobnosti.Keys)
{
    double pravdepodobnost = (double)pravdepodobnosti[key];
    nasob = nasob * pravdepodobnost;
    komb = komb * (1 - pravdepodobnost);

    Debug.WriteLine(index + " " + pravdepodobnosti[key] + " " + key );

    if (++index > listAtributu.PocetZajimavychSlov)
        break;
}

return nasob / (nasob + komb);
```

Obrázek 3.3: Algoritmus naivního Bayesovského klasifikátoru

3.2.4 Form - GUI

obsluha GUI (viz. obr.3.4) je velice snadná. dá se popsat v několika krocích:

- Stiskněte tlačítko *"Načti testovací data"* pro načtení korpusů s hamem a spamem
- Můžete testovat předem připravené zprávy (stisknutím jednoho z tlačítek *"Testovací zpráva"*) nebo napsáním vlastního textu do text boxu zadáte vlastní text pro testování. Po zadání textu jen stisknete tlačítko *"Test Box Text"* a počkáte na vyhodnocení (zobrazí se jako zpráva *"score: pravděpodobnost"* na první řádce text boxu)
- Pro výpis pravděpodobností do souboru *out.txt* stačí stisknout tlačítko *".ToFile()"*
- Pro načtení pravděpodobností ze souboru *out.txt* stačí stisknout tlačítko *".FromFile()"*



Obrázek 3.4: Grafické uživatelské rozhraní

4 Závěr

Problém filtrování spamu mi přišel velmi zajímavý. Přístup Paula Grahama jsem si zvolil z důvodu, že přestože se jedná o celkem snadno implementovatelný algoritmus, je překvapivě přesný. Přibližně 99,5% zpráv se určí správně jako spamové zprávy a pouze 0,03% z nich jsou "false positive"- tedy nespamové e-mail vyhodnocené jako spam při dostatečně velké učící množině. Moje implementace má nižší přesnost, a to z důvodu velikosti korpusů. Ve svých textech Paul Graham uvádí, že používá až několik tisíc e-mailů pro dobré i špatné testovací množiny zatímco moje implementace používá pouze několik stovek. Přesto takovéto množství by mělo mít přesnost vyšší než 90%.

Řešení bylo testováno pouze na anglicky psaných e-mailech vzhledem k jejich snadnější dostupnosti.