Detekce spamu v emailové komunikaci

František Šumšal

17. prosince 2017

1 Zadání

Cílem projektu je implementace programu pro detekci spamu pomocí offline metod.

2 Metody detekce spamu

Vzhledem k omezení ze zadání je třeba vyloučit všechny metody, které potřebují připojení k Internetu – tzn. *SPF* [1], *DKIM* [2], *DMARC* [3], které ke své funkčnosti vyžadují komunikaci s DNS serverem, detekce nebezpečných URL, různé typy blacklistů, kontrola PTR záznamů, a další.

Po vyloučení těchto metod nám zůstavají převážně metody, které zpracovávají vlastní text emailu a na jeho základě rozhodují o klasifikaci. Podskupinou těchto metod jsou metody založené na strojovém učení, a právě na jedné z těchto metod – Bayesovo filtrování – je založeno filtrování v tomto projektu.

3 Bayesovo filtrování (*Naive Bayes spam filtering*)

Základní myšlenkou Bayesova filtrování je pravděpodobnost výskytu určitých slov ve vyžádaných (*ham*) a nevyžádaných (*spam*) zprávách. Tyto pravděpodobnosti filtr nezná a je třeba je naučit pomocí předklasifikovaných zpráv.

Učení je první úskalí tohoto filtrování – je třeba získat vhodný vzorek *ham* a *spam* zpráv. Kvalita a vhodnost tohoto vzorku závisí na mnoha faktorech – velikost, jazyk, různorodost, atd. Vhodnost vzorku se liší od prostředí, kde se bude konkrétní filtr využívat. Díky těmto faktorům se poté liší úspěšnost vlastního filtru. Pro *ham* zprávy byl využit archív testovacích zpráv z internetu a pár dalších mailing listů (*Studinfo* naší fakulty a *systemd-devel* mailing list projektu *systemd*). Pro *spam* zprávy byl opět využit testovací archív zpráv z internetu a pár dalších zpráv z mého mail serveru. Výsledkem byla téměř 95% úspěšnost na školním emailu a 65% na mém osobním emailu. Pro zvýšení úspěsnosti by bylo nutné filtr nechat pár týdnů sledovat příchozí zprávy v daném prostředí, aby jeho vzorek co nejvíce odpovídal přijímanému typu zpráv.

Dalším problémem je odstranění slov či řetězců, které by mohly mít negativní vliv na filtrování. Mezi tato slova patří tzv. *stop words*, což jsou nejpoužívanější slova daného jazyka. Dalším typem řetězců k odstranění mohou být například HTML tagy v netextových emailech.

4 Implementace

Celý program je implementován v jazyce Python, verze 3. Mimo standardních knihoven je využito i několik dalších knihoven pro zpracování emailů ve formátu *eml* a pro práci s jazykem. Tyto knihovny (včetně jejich dalších závislostí) jsou nainstalovány pomocí utility `pip` při spuštění příkazu `make`.

[1] Sender Policy Framework
[2] Domain Keys Identifies Mail
[3] Domain-based Message Authentication, Reporting and Conformance

4.1 email

email je knihovna poskytována společně s instalací jazyka Python 3 a slouží pro zpracování emailů ve formátu *eml*. Z této knihovny jsou využity funkce pro načtení *eml* souboru a vyparsování potřebných částí (předmět, tělo emailu).

4.2 NLKT (Natural Language Toolkit)

Knihovna NLKT obsahuje nástroje pro práci s jazykem, např. funkci `word_tokenize`, pro rozdělení řetězce na jednotlivá slova, nebo třídu `NaiveBayesClassifier`, která obsahuje metody pro klasifikaci pomocí Bayesova filtrování. Knihovna také obsahuje databázi *stop words*. Tato databáze (včetně dalších nezbytných databází) je inicializována v souboru `Makefile`.

4.3 pickle

Pro zrychlení načítaní aplikace je využita serializace objektu Bayesova klasifikátoru pomocí knihovny pickle. Ta vezme objekt, serializuje jej a uloží do zadaného souboru. Tento objekt se dá při příštím spuštění deserializovat a znovu použít.

Po načtení a inicializaci nezbytných knihoven je inicializován objekt třídy `NaiveBayes`. Tato třída se pokusí načíst serializovaný objekt klasifikátoru ze souboru `.classifier`. Pokud neuspěje, je zahájeno strojové učení ze zadaných adresářů `spamdir` a `hamdir`. Z těchto adresářů jsou rekurzivně načteny všechny soubory, rozparsovány pomocí knihovny *eml*, rozděleny na tokeny s patřičným typem (spam/ham) a poté předány třídě `NaiveBayesClassifier`, která z těchto dat vytvoří klasifikátor. Tento klasifikátor je poté serializován a uložen do souboru `.classifier`.

Po inicializaci klasifikátoru proběhne zpracování souborů, které byly předány jako argumenty. Každý email je rozparsován dle stejného postupu jako v případě učení, jen je místo metody `train` využita metoda `classify`, která na základě naučených dat rozhodne o typu zadaného emailu. Tato informace je poté vypsána na standardní výstup ve tvaru:

```
path_to_an_email: CLASSIFICATION
```

kde `CLASSIFICATION` může být `OK`, pokud se jedná o vyžádaný email, `SPAM` v případě nevyžádaného emailu a `FAIL` v případě chyby při zpracovávání souboru (neexistující soubor, chybná struktura emailu, atd.).

5 Závěr

Výsledkem projektu je funkční implementace klasifikace emailů pomocí Bayesova filtrování. Jak bylo zmíněno výše, úspěšnost tohoto filtrování silně závisí na vzorku, který je použit k učení algoritmu, proto vzorek, který je součástí odevzdaného řešení může a nemusí fungovat s předloženými daty ke klasifikaci.

Dále by se tato klasifikace dala optimalizovat s využitím *stematizace* (namísto původního slova se klasifikuje na základě jeho kořene) a *lematizace* (podobná funkce jako u stematizace, ale místo kořene hledá základní tvar slova). Využití těchto metod zvyšuje úspěšnost výsledného filtru, ale vyžaduje další závislosti (databáze slov pro lematizaci) a zvyšuje časovou náročnost klasifikátoru.