Spam Filtr

Vojtěch Sýkora, Miroslav Falcmann

Leden 2021

Tato úloha spočívala v tom, že jsme z ničeho vytvořili dobře fungující spam filter na emaily, přičemž cílem bylo také co nejnižší časová prodleva mezi zapnutím filtru a dostáním výsledku.

1 Popis Principu

Před prácí s emaily jsme si vypsali dříve naučené informace z csv souborů nacházejících se ve složce knowledge, do seznamu slovníků, se kterými jsme poté pracovali. Vybrali jsme si raději pracovat takto spíše než pokaždé otevírat soubor a přepisovat celý soubor, což by bylo více časově náročné.

1.1 train()

Pro začátek bylo důležité si normalizovat email a vyextrahovat si z něj důležité informace například hlavičky jako byl například Subject. Toto jsme provedli ve třech krocích.

- První bylo vypsání hlaviček do slovníku headers_dict, kde jako klíč jsme dali hlavičku (např. Subject), a jemu odpovídající hodnota byl text vypsaný vpravo od té hlavičky. Pokud nějaký řádek nezačínal hlavičkou, tak jsme jej připojili k předchozímu řádku.
- Zadruhé jsme odstranili z emailu hlavičky, které už jsme měli uložené ve slovníku v paměti.
- Zatřetí nějaké emaily obsahovaly takzvané HTML tagy, které jsme námi vymyšleným algoritmem odstranili, přičemž opravdový text emailu zůstal netknutý.

Nyní jsme měli 3 hlavní způsoby rozhodování. Podle slov, speciálních znaků a také podle slov v Subjectu. Všechny tři jsme si rozdělili na správné kusy, znormalizovali, vyfiltrovali masivní listy, které vznikly na pouze to užitečné a nakonec jsme na to spustili Counter. Pomocí těchto nových vědomostí jsme si buď updateli data v csv souborech nebo přidali nová pokud nějaké slovo či znak ještě nebyl v našich csv souborech.

1.2 Náš styl zapisování vědomostí v CSV

V každém ze tří souborů máme stejné nadpisy sloupců:

- "symbol" = slovo či symbol
- "spam" = kolikrát byl "symbol" nalezen v SPAM emailu, což jsme věděli díky !truth.txt
- "ham" = kolikrát byl "symbol" nalezen v OK emailu, což jsme věděli díky !truth.txt
- "spam_perc" = $\frac{spam}{spam+ham}$

Pomocí spam_perc jsme v **.test()** hodnotili zda je email SPAM či OK.

1.3 test()

Metoda **test()** byla přímo závislá na získaných vědomostech z metody **train()**. Obdobně jako při metodě **train()** jsme si každý testovaný e-mail převedli na slovníky, kde klíče byly jednotlivá nalezená slova či symboly a hodnoty byly počty jejich výskytů. Pokud jsme nalezli slova, o kterých už máme informaci z testovací sady, použili jsme je pro výsledné rozhodnutí, zda-li je tento email spam, či není. Proto, jsme postupovali podle tohoto vzorce:

$$\frac{\sum_{i=0}^{m} spam\ perc\ slova_i}{m} = n \tag{1}$$

Kde m je počet všech slov v textu (těch, se kterými jsme se setkali během učení)

Po několika testech jsme zjistili, že filtr má nejvyšší kvalitu, když označíme e-mail jako SPAM pouze, pokud n nabývá větší hodnoty než **0,77**. Jinými slovy, pravděpodobnost spamu musí být alespoň 0,77. False positive je totiž mnohem závažnější problém, než false negative. Proto výsledné rozhodování vypadalo takto:

```
if n > 0.77:
    prediction_dict[email] = 'SPAM'
else:
    prediction_dict[email] = 'OK'
```

2 Dosažené výsledky

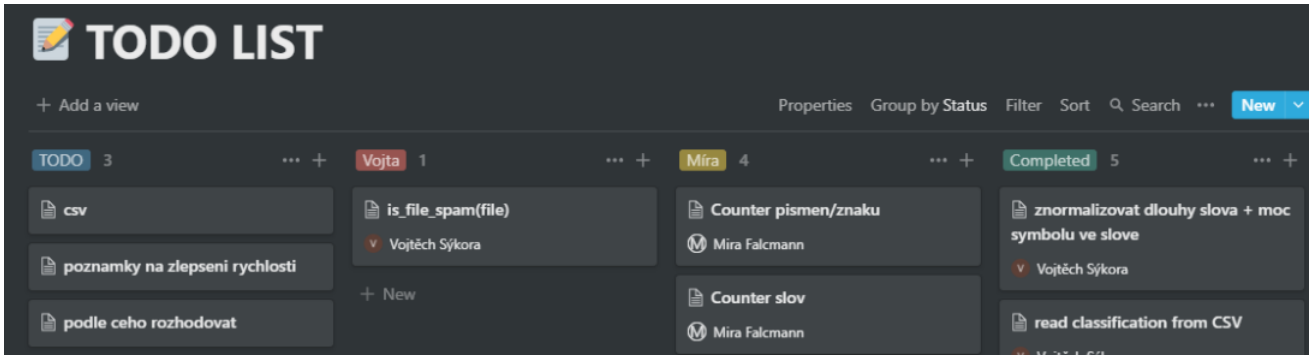
Na sadě emailů, na které jsme již trénovali se výsledky pohybovali vždy mezi 97 - 99%.

Nejdříve uchovávali hodně dat v knowledge složce, což se vymstilo kvůli dlouhému běhu programu, avšak o to přesnější byl. Když jsme si pamatovali cca 30 000 řádků informací v csv souborech, tak byla kvalita našeho filtru vždy nad 85%, avšak Brute to časově nezvládal. Poté co jsme přitvrdili na normalizaci a filtrování slov do našich csv souborů, tak se nám sice zrychloval filter ale zhoršovala kvalita. Nakonec jsme se uspokojili s kvalitou 0,915 na první sadě od Brute a 0,64 na druhé sadě od Brute.

3 Rozdělení práce a organizace týmu

Jelikož jsme na filtru pracovali ve dvojici, tak bylo namístě společné úložiště přes internet, pro což jsme si vybrali google drive. Zaprvé jsme s ním uměli hned oba dva a také bylo velmi příjemné, zapnout si automatickou synchronizaci celého projektu v počítači. Kdykoliv jeden udělal nějaké změny, tak je ten druhý mohl vidět bez jakékoli námahy.

Pro hlavní rozdělení práce a také plánování práce jsme používali Notion, což se osvědčilo jako ideální program pro tuto práci. Opravdu bych tento program doporučil všem i na jiné aktivity než skupinová práce. V Notionu jsme měli vše synchronizované a plánovali jsme si tam každý krok naší práce a kdo jej udělá. Zde je ukázka námi využívaného "Board view" na rozdělení práce:



Kde jsme si mohli přiřadit práci k člověku a také označit co je již hotovo. Levý sloupec TODO sloužil jako prostor pro jakékoliv nápady, které později prodiskutujeme a někomu přiřadíme.

3.1 Kdo co udělal

Tato tabulka je orientačně, neboť jsme si průběžně pomáhali a někdy třeba někdo vymyslel co a jak bysme mohli udělat a ten druhý to naimplementoval.

Normalizace emailů - headers_dict	VS
Normalizace emailů - leave only body	VS
Normalizace emailů - remove html tags	MF
Práce se slovy	MF
Práce se specielnímy znaky	MF
Práce se subjectem	VS
CSV - zapisovani, cteni, rozložení souboru	VS
Filtrování našich slovníků	MF, VS

4 Zhodnocení, závěr

Vývoj spam filtru byl zatím největší programovací projekt, do kterého jsme se zatím pustili. Celková práce nám zabrala zhruba pět celých dnů, během kterých jsme promysleli strategii našeho filtru, rozčlenili problém do několika dílčích metod, implementovali samotné metody a následně je složili dohromady, abychom dali vznik samotnému spam filtru.

Během této doby, jsme byli v častém kontaktu a vymýšleli různá zlepšení, která by zvedla kvalitu našeho filtru.

Přesto, že práce bylo mnoho, tak nás práce bavila a rádi jsme se samotnému projektu věnovali.

Seznam použité literatury, online zdrojů

- https://www.w3schools.com/
- https://cw.fel.cvut.cz/