Warning

This page is located in archive. Go to the latest version of this course pages. Go the latest version of this page.

Spam filter - krok 6: Vlastní filtr

Implementujte vlastní, co nejlepší filtr.

Testy [/b211/courses/b4b33rph/cviceni/spam/unit_testing] ke kroku 6:

- samostatně test6_filter.zip [/b211/_media/courses/b4b33rph/cviceni/spam/test6_filter.zip] nebo
- společně se všemi předchozími testy test6_all.zip [/b211/_media/courses/b4b33rph/cviceni/spam/test6_all.zip].

Příprava

Rozmyslete si, jaké vnitřní mechanismy byste chtěli ve svém filtru použít, podle čeho by se váš filtr měl rozhodovat mezi korektním emailem a spamem a jak by se dal takový filtr učit z dat.

Projděte si také sekci s několika tipy, které by se vám při konstrukci filtru mohly hodit.

Programovací tipy

Na tomto místě bychom vám rádi poskytli pár užitečných tipů ke konstrukci vlastních filtrů. Zdaleka ne vše, co je v této sekci uvedeno, budete nutně potřebovat. Např. nepředpokládáme, že budete využívat modul `email` nebo regulární výrazy (i když samozřejmě můžete).

Tokenizace

Čítače

Procházení uspořádaných prvků slovníku (čítače)

Tokenizace pro pokročilé aneb regulární výrazy

Struktura mailu

Třída "MyFilter"

Úkol:

- Vytvořte vlastní funkční spam filtr.

K čemu nám to bude:

- Budete jej odevzdávat a budete za něj hodnoceni.

Následující specifikace pro třídu `MyFilter` jsou povinné a musíte jim vyhovět!

Specifikace

V modulu `filter.py` vytvořte třídu `MyFilter`, která bude schopna

- převzít trénovací korpus při zavolání metody `train(train_corpus_dir)`,
- naučit filtr na trénovacích emailech (pokud se tak rozhodnete) a
- ohodnotit všechny emaily testovacího korpusu při zavolání metody `test(test_corpus_dir)`, tj. vytvoří v adresáři testovacího korpusu soubor `!prediction.txt`. Metoda by měla být schopná pracovat i tehdy, když předtím nedojde k natrénování filtru metodou `train()`.
- Filtr má za úkol detekovat spamy, očekává se tedy, že **SPAMy bude považovat za pozitivní** příklady a **HAMy za negativní**, detaily viz krok 3 [/b211/courses/b4b33rph/cviceni/spam/krok3].
- Vnitřní mechanismy vašeho spam filtru jsou zcela na vás!
- **Časový limit**: Učení filtru a ohodnocení testovacího korpusu by celkem nemělo zabrat více než 5 minut; filtru by mělo stačit i výrazně méně času.

Použití třídy MyFilter

Typické použití třídy `MyFilter` bude následující:

```
from filter import MyFilter

filter = MyFilter()
filter.train('/path/to/training/corpus')  # Tento adresář bude obsahovat soubor !t
filter.test('/path/to/testing/corpus')    # V tomto adresáři metoda vytvoří soubor
```

Protože by ale metoda `test()` měla být schopná pracovat i bez předchozího volání metody `train()`, je přípustné i toto použití (a bude také testováno):

```
from filter import MyFilter
```

```
filter = MyFilter()
filter.test('/path/to/testing/corpus')    # V tomto adresáři metoda vytvoří soubor
```

Při hodnocení kvality predikcí filtru samozřejmě budeme vždy volat metodu `train()` před metodou `test()`.

Trénování

- Pokud se váš spam filtr nebude umět učit, můžete trénovací corpus ignorovat. Pokud jste se rozhodli držet našich doporučení, měli byste ve třídě `MyFilter` implementovat metodu `train()`, která ovšem může být prázdná.
- V trénovacím korpusu zcela jistě bude soubor `!truth.txt` s informací o skutečné třídě emailů v daném korpusu.

Testování

- Váš filtr **musí** v adresáři testovacího korpusu **vytvořit soubor** `!prediction.txt`, v němž pro každý soubor s emailem v testovacím korpusu uvede řádek `nazevsouboru OK`, jedná-li se (podle mínění filtru) o korektní email, nebo `nazevsouboru SPAM`, jedná-li se (podle mínění filtru) o spam.
- *Váš filtr nesmí při testování v žádném případě využívat soubor `!truth.txt` v adresářích testovacích korpusů. Testy budou probíhat tak, že tam žádný takový soubor nebude!*

Možné varianty filtrů

1. Natvrdo "zadrátovaný" filtr, který se neumí učit. Takový filtr bude mít patrně prázdnou metodu `train()`. V metodě `test()` pak bude nějakým způsobem využívat postupy zvolené autorem, podle nichž pro každý testovací email určí, jedná-li se o spam nebo ham.
2. Filtr využívající předem připravené externí informace (např. předem naučený filtr). Filtr bude mít asi také prázdnou metodu `train()`. V metodě `test()` se pak bude rozhodovat na základě externích informací uložených v souboru (souborech) uploadovaných spolu se zdrojovým kódem filtru. Důvodem k takovéto "architektuře" může být např. to, že proces učení filtru je časově náročný - filtr byl studentem naučen offline a předem. (Pokud chce student v tomto případě body za to, že se filtr umí učit, musí na cvičení svého cvičícího přesvědčit, že předem připravené informace skutečně pocházejí z procesu učení.) Jiným důvodem použití této varianty může být využití nějakého externího slovníku, apod.
3. Učicí se filtr. Filtr nemá prázdnou metodu `train()` a ta je dostatečně rychlá [/b211/courses/b4b33rph/cviceni/spam/specifikace#casove_pozadavky], že zvládne extrahovat potřebné znalosti z informací uložených v řádově stovkách emailů (velikost trénovací sady).