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 4

Vytvořte 3 jednoduché neadaptivní filtry, paranoidní, naivní a náhodný, a u každého z nich určete, jak kvalitní predikce poskytuje.

Testy [/b211/courses/b4b33rph/cviceni/spam/unit_testing] ke kroku 4:

- samostatně test4_simplefilters.zip [/b211/_media/courses/b4b33rph/cviceni/spam/test4_simplefilters.zip] nebo
- společně se všemi předchozími testy test4_all.zip [/b211/_media/courses/b4b33rph/cviceni/spam/test4_all.zip].

Příprava

Rozmyslete si a načrtněte na kus papíru:

- Jakým způsobem se spam filtr vlastně používá?
- Jaký je z hlediska implementace rozdíl mezi učicím se filtrem a filtrem, který se učit neumí?
- Existuje nějaká část, kterou budou mít všechny spam filtry společnou?
- Je lepší realizovat spam filtr jako funkci nebo jako objekt s vlastnostmi a metodami?
- Jaké jsou minimální požadavky na tuto realizaci? Co všechno by měla umět? Jaké musí mít vstupy a co musí být jejím výstupem?

Příprava: Dědičnost

Prostudujte si, jak v OOP funguje a jak se v Pythonu realizuje *dědičnost*. Informace najdete např.

- v oficiálním Python tutoriálu [https://docs.python.org/3.4/tutorial/classes.html#inheritance], nebo
- v [Wentworth2012], kapitola 23 [http://openbookproject.net/thinkcs/python/english3e/inheritance.html] (navazuje na kapitolu 22 [http://openbookproject.net/thinkcs/python/english3e/collections.html]).

Jednoduché filtry

Úkoly:

- **V modulu** `simplefilters.py` vytvořte tři třídy představující tři hloupé filtry:
  - `NaiveFilter`, který bude všechny emaily klasifikovat jako OK,
  - `ParanoidFilter`, který bude všechny emaily klasifikovat jako SPAM, a
  - `RandomFilter`, který bude maily náhodně zařazovat do tříd OK a SPAM.
  - Mají-li tyto filtry nějaké části společné, zkuste tyto rysy extrahovat do společného předka `BaseFilter` **v modulu** `basefilter.py`.

K čemu nám to bude:

- Na jednoduchých filtrech si ukážeme základní kostru třídy představující v našem "frameworku" spamový filtr. Současně budeme mít k dispozici nějaké filtry, jejichž kvalitu si budeme moci otestovat pomocí funkcí z předchozího kroku.

Specifikace

> Abychom umožnili pozdější automatické testování vašeho finálního filtru, budeme vyžadovat, aby se třída vašeho filtru jmenovala `MyFilter` a byla umístěna v modulu `filter.py`. V tomto kroku ale máte vytvořit 3 třídy pojmenované `NaiveFilter`, `ParanoidFilter` a `RandomFilter` umístěné v modulu `simplefilters.py`.

Filtr pro nás bude představován třídou, která bude mít minimálně dvě metody: `train()` a `test()`. Filtry, které se nebudou učit z dat, budou mít metodu `train()` prázdnou. Další struktura třídy je libovolná.

Metoda `train()`:

Vstupy	Cesta k adresáři s ohodnocenými emaily, tj. adresář musí obsahovat soubor `!truth.txt`. (Pro jednoduché filtry je to jedno.)
Výstupy	Nic.
Efekty	Vytvoření a nastavení vnitřních datových struktur třídy, aby byly později využitelné metodou `test()`.

Metoda `test()`:

Vstupy	Cesta k adresáři s maily. (Adresář nebude obsahovat soubor `!truth.txt`.)
Výstupy	Nic.

Efekty	Vytvoří v zadaném adresáři soubor `!prediction.txt` .

Hodnocení kvality jednoduchých filtrů

Vytvořte jednoduchý skript, který vypočte kvalitu predikcí zvoleného filtru. Skript:

- importuje třídu zvoleného filtru,
- zavolá metodu `train()` na jedné poskytnuté datové sadě,
- zavolá metodu `test()` na druhé poskytnuté datové sadě,
- zavolá `compute_quality_for_corpus()` pro druhou datovou sadu,
- kvalitu vypíše,
- odstraní soubor `!prediction.txt` z datové sady.

courses/b4b33rph/cviceni/spam/krok4.txt · Last modified: 2018/07/17 13:25 (external edit)