**Спам филтър**

Изготвили

Владимир Кузмов, ф.н. 25535, ИИОЗ

Тихомир Кожухарски ф.н. 25387, ИИ

02.06.2017

Съдържание

[Декларация за липса на плагиатство 3](#__RefHeading__1701_435126117)

[1. Мотивация 4](#__RefHeading__1483_435126117)

[2. Кратък обзор 4](#__RefHeading__1497_435126117)

[3. Нашето решение 5](#__RefHeading__1485_435126117)

[4. Програмна реализация 7](#__RefHeading__1487_435126117)

[5. Резултати от експерименти 8](#__RefHeading__1489_435126117)

[6. Заключение 8](#__RefHeading__1491_435126117)

[Използвана литература и ресурси: 9](#__RefHeading__1493_435126117)

# Декларация за липса на плагиатство

- Тази курсова работа е наша работа, като всички изречения, илюстрации и програми от други хора са изрично цитирани.

- Тази курсова работа или нейна версия не са представени в друг университет или друга учебна институция.

- Разбираме, че ако се установи плагиатство в работата ни ще получим оценка “Слаб”.

- Трите имена и подпис на студентите:

# 1. Мотивация

Спам съобщенията са наболял проблем, с който много компании се борят от десетки години, но все още няма решение, което да гарантира 100% успеваемост при разпознаването на това дали едно съобщение е спам или не. Те са нежелани електронни съобщения, които са най-често с рекламно съдържание. Най-често, спам съобщенията имат следните форми:

* писма, изпратени на електронната поща
* кратки съобщения на мобилните устройства (SMS)
* коментари в блогове
* коментари във форуми

В тази курсова работа ще разгледаме по-конкретно проблема със спам email-ите.

Получаването на спам пречи на нормалната работа с изброените електронни услуги, тъй като потребителите трябва да отсеят ръчно спам имейлите от останалите.

Целта на нашият проект е да представим различни подходи за автоматична класификация на имейли като спам (spam) или не спам (ham), използвайки подходи от машинното самообучение. За целта ще бъдат разгледани множество класификатори и ще бъдат сравнени резултатите чрез различни метрики.

# 2. Кратък обзор

Има много техники, които се прилагат в борбата със спама. За съжаление няма техника, която напълно да решава проблема със спама. Всяка техника има своите предимства и недостатъци и се опитва да постигне баланс, с който да се избегне неправилното отхвърляне на легитимни имейли (false positive) и пропускане на спам имейли (false negative). Анти-спам техниките могат да бъдат разбити на големи категории:

* техники, които изискват действие от страна на някой човек
* техники, които могат да бъдат автоматизирани от администратори на email услугите
* техники, които могат да бъдат автоматизирани от хората, които изпращат съобщенията
* техники, които се прилагат от правоохранителните органи

В повечето случаи хората са по-склонни да получат някой спам имейл в повече, пред това да се блокира легитимен имейл. Откриването на добър баланс между false positive и false negative е от съществено значение за една анти-спам система да бъде успешна.

Една обещаваща техника за борба със спам е поддръжката и използването на DNS черни списъци (DNSLB – DNS Blacklist). Те представляват списъци от домейн имена и IP адреси на сървъри от следните типове:

* такива, за които се знае, че изпращат спам или се знае, че са имейл сървъри,
* имейл сървъри, които не изискват автентикация при изпращането на имейли
* прокси сървъри
* други адреси, за които се знае, че не би трябвало да изпращат имейли (например адреси, заделени за IP адреси на крайни клиенти на интернет доставчици)

Използването на такива черни списъци води до постигането на много добри резултати.

Много често се използват техники за засичане на спам, базирани на съдържанието на имейла, като например засичането на ключови думи, от сорта на “viagra” или с използването на статистически методи. Такъв тип техники биха могли да са доста точни, ако са настроени правилно спрямо вида на легитимните имейли, които се очаква да се получават. От друга страна, биха могли да допускат грешки, като например, засичането като спам дума на низа “cialis” в думата “specialist”. Хората, които пращат спам много често се опитват да прескочат такива бариери, като използват техники като:

* замяната на букви, с техния вариант с ударение или пък с алтернативна буква, която се изобразява по същия начин, но има различно вътрешно представяне (например подмяната на латинската буква “A” с нейния вариант на кирилица “A”)
* вмъкването на други символи между буквите, като интервали или други символи, които не се принтират, за да могат да попречат разпознаването по шаблони.

Поради тази причина, за автоматично разпознаване на спам имейли се използват не само прости шаблони, които биха могли лесно да бъдат заблудени, а и методи от машинното самообучение, които да откриват по-сложни признаци за това дали едно съобщение е спам или не. Използват се класификационни модели, които са предварително тренирани, за двоично разпознаване. В пакетите на някои имейл сървърни системи се съдържат модели, които са предварително тренирани върху големи обеми от имейли. Голямо предимство е, че моделите биха могли лесно да бъдат тренирани върху наличните имейли, за да могат да се приспособят към предметната област на съобщенията, които се получават на конкретния имейл сървър.

# 3. Нашето решение

Решихме да разгледаме подробно представянето на някои от най-известните алгоритми за класификация, познати от машинното самообучение, при използването им за класификация на имейл съобщения като спам или не-спам (ham). За тренирането на моделите използвахме колекцията от спам и легитимни имейли “Enron”.

## 3.1. Features

Направихме изследвания със следните видове feature-и:

* Feature-и, базирани на символи:
  + Броя на символите в съдържанието на съобщението
  + Съотношението на броя на буквите, към общия брой символи в съобщението
  + Съотношението на броя на цифрите, към общия брой символи в съобщението
  + Съотношението на броя на интервалите, към общия брой символи в съобщението
  + Съотношението на броя на специалните символи, към общия брой символи в съобщението
* Feature-и, базирани на думи:
  + Bag-of-words – представяме съдържанието на имейлите като вектор от честотата на срещанията на 3000-те най-често срещани думи в корпуса
  + Брой думи в съдържанието
  + Съотношението на броя на кратките думи (с дължина по-малка или равна на 2), към общия брой на думите в съдържанието
  + Средната дължина на думите
  + Съотношението на броя на уникалнитедуми, към общия брой на думите в текста
  + Броя на уникалните спам думи
  + Булева наличност на спам думи
* Ниво на сложност на текста (например – 6-ти клас, 12-ти клас и т.н.)

След като съберем всички feature-и за всички обекти от колекцията, ние ги нормализираме, за да можем да елиминираме негативното въздействие на екстремални стойности върху класификационната точност.

## 3.2. Класификатори

За нашите изследвания, решихме да използваме следните класификатори:

* SVM (linear kernel)
* SVM RBF
* TwinSVM
* Multinomial Naive Bayes
* KnearestNeighbours
* Decision Tree
* Random Forest
* Artificial Neural Network
* Linear Discriminant Analysis
* Quadratic Discriminant Analysis

## 3.3. Използвани метрики за оценяване

Оценихме качеството на класификация на избраните модели, със следните метрики:

* Precision
* Recall
* F1
* Accuracy

# 4. Програмна реализация

## 4.1. Среда за разработка и използвани библиотеки

Поради огромното количество налични библиотеки за машинно самообучение за Python, решихме да го използваме точно него като програмен език, с който да направим нашите изследвания. Приложението може да бъде стартирано с Python 3.x. Следва списък от използваните допълнителни библиотеки и съответните им версии:

* sklearn 0.18.1
* scipy 0.19.0
* tensorflow 1.1.0
* numpy 1.12

## 4.2. Разработени скриптове

Реализирахме спам филтъра като набор от python скриптове, състоящ се от следните модули:

* pipeline.py - това е главният модул, който има следните функции:
  + Зарежда данните от Enron колекцията
  + Разбива данните на тренировъчни и тестови, в съотношение 3:2
  + Трансформира всеки мейл от заредените данни във вектор от feature-и, използвайки избраните feature модули. По този начин, всички имейли от колекцията се преобразуват във feature матрица и отделен вектор с целевите стойности.
  + Тренира избраните модели, с получената feature матрица и вектора с целевите стойности.
  + Проверява и принтира точността на всеки от избраните модели, използвайки вече описаните метрики, с подготвените тестови данни.
* model.py – съдържа функции, които използваме за преобразуването на събраните feature-и във feature матрица
* twinsvm – пакет (директория), предоставящ имплементация на TwinSVM, който е свален от тук: <https://github.com/AK101111/Twin-SVM>.
* features – пакет, съдържащ класовете, които извличат feature-ите, които използваме
  + character\_based – пакет с класове за излвличане на feature-и, основани на символи
    - alpha\_ratio.py – модул, съдържащ класа, който извлича feature-а за съотношението на броя букви към общия брой символи
    - digit\_ratio.py – модул, съдържащ класа, който извлича feature-а за съотношението на броя цифри към общия брой символи
    - special\_chars\_ratio.py – модул, съдържащ класа, който извлича feature-а за съотношението на броя специални символи към общия брой символи
    - whitespace\_ratio.py – модул, съдържащ класа, който извлича feature-а за съотношението на броя интервални символи към общия брой символи
    - number\_of\_characters.py – модул, съдържащ класа, който извлича feature-а за броя на символите в имейл съобщението
  + word\_based – пакет с класове за излвличане на feature-и, основани на думи
    - average\_word\_len.py - модул, съдържащ класа, който извлича feature-а за средната дължина на думите в текста
    - number\_of\_words.py - модул, съдържащ класа, който извлича feature-а за броя на думите в имейла
    - short\_words\_ratio.py - модул, съдържащ класа, който извлича feature-а за съотношението на броя на кратките думи, към общия брой на думите в имейла
    - unique\_words\_ratio.py - модул, съдържащ класа, който извлича feature-а за съотношението на броя на уникалните думи, към общия брой на думите в имейла
    - word\_counts.py - модул, съдържащ класа, който извлича feature-ите за честотата на срещане на 3000-те най-често срещани думи в използвания корпус
    - spam\_words.py - модул, съдържащ класа, който извлича feature-а за броя на уникалните спам думи, които са намерени в имейла
    - spam\_regexes.py - модул, съдържащ всички регулярни изрази, които се използват за откриване на спам думи в имейлите. Тези регулярни изрази са взети от конфигурацията на приложението spamassassin.
  + flesch\_reading\_score.py - модул, съдържащ класа, който извлича feature-а за сложността на текста

# 5. Резултати от експерименти

Таблица 1 – резултати от експерименти проведени върху всички фичъри, без топ 3000 използвани думи (11 фичъра).

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Algorithm | Precision | Recall | F1 | Accuracy | Training time (min) | Prediction time (min) |
| SVC (linear) | 0.68 | 0.65 | 0.66 | 66.44 | 0 | 0 |
| SVC (rbf) | 0.68 | 0.84 | 0.76 | 72.25 | 0 | 0 |
| MultinomialNB | 0.59 | 0.73 | 0.65 | 59.87 | 0 | 0 |
| kNN | 0.72 | 0.73 | 0.73 | 72.06 | 0 | 0 |
| Neural Network | 0.51 | 1.0 | 0.67 | 50.92 | 0 | 0 |
| DecisionTree | 0.72 | 0.67 | 0.69 | 69.77 | 0 | 0 |
| RandomForest | 0.67 | 0.8 | 0.73 | 69.67 | 0 | 0 |
| MLPClassifier | 0.67 | 0.68 | 0.67 | 66.5 | 0 | 0 |
| AdaBoost | 0.73 | 0.71 | 0.72 | 71.68 | 0 | 0 |
| GaussianNB | 0.57 | 0.88 | 0.69 | 59.46 | 0 | 0 |
| Linear Discriminant Analysis | 0.63 | 0.81 | 0.71 | 65.89 | 0 | 0 |
| Quadratic Discriminant Analysis | 0.57 | 0.36 | 0.44 | 53.67 | 0 | 0 |
| TwinSVM | 0.64 | 0.81 | 0.71 | 66.83 | 46 | 0 |
| TwinSVM with kernel | 0.63 | 0.83 | 0.72 | 66.46 | 128 | 26 |

Таблица 2 – резултати от експерименти проведени върху фичъри топ 3000 използвани думи (3000 фичъра).

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Algorithm | Precision | Recall | F1 | Accuracy | Training time (min) | Prediction time (min) |
| SVC (linear) | 0.87 | 0.99 | 0.93 | 92.23 | 16 | 10 |
| SVC (rbf) | 0.97 | 0.77 | 0.86 | 87.17 | 49 | 11 |
| MultinomialNB | 0.97 | 0.95 | 0.96 | 95.98 | 1 | 0 |
| kNN | 0.85 | 0.95 | 0.9 | 89.09 | 7 | 19 |
| Neural Network | 0.98 | 0.98 | 0.98 | 97.89 | 15 | 0 |
| DecisionTree | 0.75 | 0.98 | 0.85 | 82.75 | 0 | 0 |
| RandomForest | 0.68 | 0.83 | 0.75 | 71.68 | 0 | 0 |
| MLPClassifier | 0.94 | 0.99 | 0.96 | 96.13 | 1 | 0 |
| AdaBoost | 0.95 | 0.96 | 0.95 | 95.31 | 1 | 0 |
| GaussianNB | 0.92 | 0.96 | 0.94 | 93.94 | 0 | 0 |
| Linear Discriminant Analysis | 0.88 | 0.91 | 0.89 | 89.11 | 1 | 0 |
| Quadratic Discriminant Analysis | 0.98 | 0.93 | 0.95 | 95.52 | 1 | 0 |
| TwinSVM | 0.96 | 0.99 | 0.97 | 97.32 | 59 | 0 |
| TwinSVM with kernel | 0.96 | 0.99 | 0.97 | 97.23 | 247 | 62 |

# 6. Заключение

Оказва се, че при използване на по-сложни фичъри като word based и character based, описани по-горе, резултатите не са много задоволителни. Нещо доста по-просто като най-често срещаните думи върши значително по-добра работа. Тъй като 3000 е сравнително голям брой фичъри, се вдига времето за трениране на моделите, както и времето за предсказване, но това е за сметка на почти 1.5 пъти по-голяма точност на предсказването.

# Използвана литература и ресурси:

1. Енрон дейтасет - <http://www.cs.cmu.edu/~enron/>
2. Какво е спам - <https://en.wikipedia.org/wiki/Spamming>
3. Спам техники - <https://en.wikipedia.org/wiki/Anti-spam_techniques>
4. Flesch-Kincaid reading score - <https://en.wikipedia.org/wiki/Flesch%E2%80%93Kincaid_readability_tests>
5. Какво е Twin SVM - <http://www.sciencedirect.com/science/article/pii/S1110866514000498>
6. Twin SVM за python - <https://github.com/AK101111/Twin-SVM>
7. SVM базиран спам филтър с активно обучение - <http://trec.nist.gov/pubs/trec15/papers/hit.spam.final.final.pdf>
8. Спам филтър с невронни мрежи - <https://arxiv.org/pdf/1502.03581.pdf>

Принос на Владимир Кузмов:

* Изготвяна на две презентации
* Имплементация на невронна мрежа на python (neural\_network.py)
* Имплементация на фичър flesch\_reading\_score (features/flesh\_reading\_score.py)
* Документация:
  + Извеждане на таблиците с резултатите
  + Заключителна част
  + Референции

Принос на Тихомир Кожухарски:

* Документация
  + Мотивация
  + Кратък обзор
  + Нашето решение
  + Програмна реализация
* Имплементация на character\_based и word\_based фичъри на python
* Имплементация на pipeline на python (pipeline.py)