МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
СХІДНОУКРАЇНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ВОЛОДИМИРА ДАЛЯ
ФАКУЛЬТЕТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ТА ЕЛЕКТРОНІКИ
КАФЕДРА ПРОГРАМУВАННЯ ТА МАТЕМАТИКИ

Пояснювальна записка

до дипломної роботи

бакалавр

(освітньо-кваліфікаційний рівень)

на тему «Визначення спам-зображень за допомогою перцептивних хешів»

Виконав: групи ІТ-151
напряму підготовки 6.040302 «Інформатика»
_____ Папірний Д.М.
(підпис)

Керівник,
доцент, к.т.н._____ Іванов В. Г.
(підпис)

Рецензент,
_____ Батурін О.І.
(підпис)

СЄВЄРОДОНЕЦЬК
2019 року

СХІДНОУКРАЇНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ВОЛОДИМИРА ДАЛЯ

Факультет інформаційних технологій та електроніки
Кафедра програмування та математики
Освітньо-кваліфікаційний рівень бакалавр
Спеціальність **6.040302 «Інформатика»**

ЗАТВЕРДЖУЮ
Завідувач кафедри ПМ,
д.т.н., доцент
_____Лифар В.О.
«___» _____ 2019 р.

З А В Д А Н Н Я
НА ДИПЛОМНУ РОБОТУ СТУДЕНТУ
ПАПІРНИЙ ДМИТРО МИКОЛАЙОВИЧ

1. Тема роботи Визначення спам-зображень за допомогою перцептивних хешів керівник роботи доцент Іванов Віталій Геннадійович
2. Строк подання студентом роботи 06 червня 2019 р.
3. Вихідні дані до роботи

   Об'єктом дослідження даної роботи є програмне забезпечення пошуку спам-зображень.

   3.1 Літературні джерела:

   Egmont-Petersen, M., de Ridder, D., Handels, H. "Image processing with neural networks - a review". Pattern Recognition 35 (10), pp. 2279–2301(2002)

   Christoph Zauner. Implementation and Benchmarking of Perceptual Image Hash Functions (2010)

   D. Marrand, E. Hildret. Theory of edge detection, pp. 187-215 (1979)

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)

   4.1 Вступ

   4.2 Огляд методів:

   Перцептивні хеш-алгоритми.

   Simple Hash

   Discrete Cosine Transform Based Hash

   4.3 Основна частина, в якої висвітлити:

   Засоби реалізації

   Пристрій спам-фільтра

   4.4 Висновки

   4.5 Перелік використаних джерел
5. Перелік графічного матеріалу немає
6. Дата видачі завдання 02 лютого 2019 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломної роботи	Строк виконання етапів роботи	Примітка
1	Одержання завдання на виконання роботи	01.02.19	
2	Укладання і погодження з керівником плану і етапів виконання роботи	20.02.19	
3	Узагальнення даних літературних джерел, укладання першого розділу	1.03.19	
4	Аналіз шляхів виконання завдання. Вибір і погодження з керівником оптимального шляху	10.03.19	
5	Проектування інфологічної моделі задачі що реалізується.	01.04.19	
6	Укладання та тестування програмного продукту	20.04.19	
7	Укладання, оформлення та погодження пояснювальної записки з керівником	15.05.19	
8	Здача готової пояснювальної записки на кафедру	06.06.19	
9	Укладання доповіді і презентації	10.06.19	

Студент _____ Папірний Д.М.
(підпис)

Керівник роботи _____ Іванов В. Г.
(підпис)

ЛИСТ ПОГОДЖЕННЯ І ОЦІНЮВАННЯ

дипломної роботи студента гр.ІТ-151 Папірного Д.М.

Науковий керівник

Доцент, к.т.н. _____ Іванов В. Г.

Оцінка наукового керівника: _____

Рецензент ст..викл. каф. ПМ СНУ ім.В.Даля Батурін О.І.
місто роботи, посада, ПІБ

Оцінка рецензента: _____

Кінцева оцінка за результатами захисту:
_____

Голова ЕК

_____ Лифар В.О.
підпис

Ми живемо в епоху інформаційних технологій, коли комп'ютер є невід'ємною частиною нашого життя, а Інтернет необхідний для роботи, навчання, та просто для життя. Дуже часто доводиться стикатися з проблемою спаму в Інтернеті.

Спам є небажаною, настирливої, а іноді і шкідливої інформації. Спам здається нешкідливим, але це тільки на перший погляд. На самому справі, хвиля нав'язаної реклами засмічує поштові скриньки, соціальні мережі, сайти знайомств, месенджери і блоги і змушує користувачів витрачати гроші на оплату надлишкового трафіку, а також особистий час на «чистку» від спаму. Частка спаму в світовому поштовому трафіку становить близько 80% на 2011 рік [1]. Також спам найчастіше використовується як знаряддя шахраїв для виманювання грошей.

За останні роки було винайдено чимало способів боротьби зі спамом. На жаль, спамери відстежують дії фільтрів і винаходять все нові прийоми для їх обходу. До того ж, нерідко фільтрація спаму приносить більше шкоди, ніж користі: разом з настирливої рекламою не доходять до адресата і важливі ділові або особисті повідомлення.

Одним з нових і найбільш шкідливих прийомів обходу сучасних спам-фільтрів є використання спам-зображень.

Це зображення, на які накладаються так звані спам-фрагменти, що рекламують різні сайти, продукти, послуги. спам-фрагменти являють собою картинки, до яких застосовуються деякі з наступних спотворень: повороти, масштабування, затемнення, розмиття, зміна яскравості, кольорів. фрагменти зазвичай

займають певну частину від вихідного зображення. також фрагменти можуть мати прозорий фон і накладатися на різні місця на

зображенні-підкладці. Таким чином, зображення потрібно вважати спамом, якщо на ньому немає будь-якої спам-фрагмент.

В даний момент для боротьби з таким типом спаму активно використовується для користувача модерація. Коли користувач Інтернет сервісу бачить спам-зображення, він відзначає його як спам. при накопиченні порогового значення спаму, зображення видаляється з бази даних. Але такий спосіб працює погано, тому що спамери постійно змінюють підкладки і незначно спотворюють фрагмент - в результаті кожне конкретне зображення використовується відносно небагато раз і не встигає набрати потрібну кількість спам-окулярів. Для вирішення цієї проблеми ми пропонуємо використовувати автоматичний пошук і розпізнавання фрагментів спаму за допомогою алгоритмів комп'ютерної обробки зображень.

Традиційним вирішенням завдання розпізнавання зображень є використання нейронних мереж [2]. Суть цього рішення в тому, що спочатку нейронна мережа проходить навчання за допомогою навчальної вибірки, а потім їй на вхід подаються спам-зображення, які на виході повинні бути визначені як спам або не спам. Але у такого

рішення є проблема: потрібно мати велику кількість навчальних даних для кожного типу спам-зображень, що неможливо, так як новий спам може з'являтися постійно.

Рішення, яке може давати хороші результати для навчальної вибірки розміром всього кілька зображень - використання перцептивних хеш-алгоритмів [3]. У звичайному розумінні хеш-функції повертають різні значення хешів для різних даних і з великою ймовірністю однакові значення хеш-кодувань для одних і тих же даних. Для схожих даних звичайні хеш-функції можуть повертати абсолютно різні значення. А перцептивное хешування - це окремий випадок локально-чутливого хешування [4]. Тому перцептивні хеш-функції, як і локально-чутливі, для близьких об'єктів повертають близькі значення хешів. перцептивні хеші можна порівнювати між собою і робити висновки про ступінь подібності двох наборів даних. Назва «перцептивних» хеш-функцій відбулося завдяки тому, що в ході обчислення значення хешу застосовуються процеси, що імітують різні аспекти сприйняття інформації людиною [2].

Існує кілька поширених перцептивних хеш алгоритмів для роботи з зображеннями. Дана робота присвячена вивчення цих алгоритмів і їх адаптації для завдання пошуку спам-зображень.

Постановка задачі

В рамках даної роботи були поставлені наступні завдання:

 Вивчити різні існуючі підходи для обчислення перцептивних хешів зображень;

 Реалізувати існуючі хеш-алгоритми і адаптувати їх для завдання пошуку спам-зображень;

 Протестувати отримані алгоритми на базі даних з зображеннями, проаналізувати результати їх роботи (порівняти швидкість і ефективність алгоритмів);

 Підготувати умови для введення алгоритмів в експлуатацію

огляд

Існує кілька перцептивних хеш-алгоритмів для роботи з зображеннями. Зазвичай побудова хешу складається з 3 основних стадій:

 Попередня обробка. На цій стадії зображення приводиться до вигляду, в якому його легше обробляти для побудови хешу. Це може бути застосування різних фільтрів (наприклад, Гаусса), знебарвлення, зменшення розмірів зображення і т.д.

 Основні обчислення. З отриманого на 1 стадії зображення будується матриця (або вектор). Матриця (вектор) може представляти з себе матрицю частот (наприклад, після перетворення Фур'є), гистограмму яркостей, або ще більш спрощене зображення.

 Побудова хешу. З матриці (вектора), отриманої на 2 стадії беруться деякі (можливо все) коефіцієнти і перетворюються в хеш. Зазвичай хеш виходить розміром від 8 до ~ 100 байт.

Попередньо обчислені значення залишаться хешів потім порівнюються за допомогою функцій, що обчислюють «відстань» між двома хешамі.

У цьому розділі описуються деякі існуючі перцептивні хеш алгоритми і функції порівняння хешів.

3.1 Перцептивні хеш-алгоритми

Нижче розглянемо докладніше 4 перцептивних хеш-алгоритму: Simple Hash [5], DCT Based Hash [2], [12], Radial Variance Based Hash [2], [6] і Marr-Hildreth Operator Based Hash [2], [7].

Simple Hash

Суть даного алгоритму полягає в відображенні середнього значення низьких частот. У зображеннях високі частоти забезпечують деталізацію, а низькі - показують структуру. Тому для побудови такий хеш-функції, яка для схожих зображень видаватиме

близький хеш, потрібно позбутися від високих частот. кроки алгоритму наступні:

 Зменшити розмір. Найшвидший спосіб позбутися від високих частот - зменшити зображення. Зображення зменшується до розміру 32x32.

 Прибрати колір. Маленьке зображення переводиться в градації сірого, так що хеш зменшується втричі.

 Знайти середнє. Обчислити середнє значення кольору для всіх 1024 пікселів.

 Побудувати ланцюжок бітів. Для кожного пікселя робиться заміна кольори на 1 або 0 в залежності від того, більше він або менше середнього. (Рис. 3.2)

a) Початкове зображення b) Отриманий «відбиток»

Малюнок 3.1 Приклад застосування алгоритму simple hash

 Побудувати хеш. Переклад 1024 бітів в одне значення. порядок не

має значення, але зазвичай біти записуються зліва направо, зверху вниз.

Отриманий хеш розміром 128 байт стійкий до масштабування, стиску або розтягування зображення, зміни яскравості, контрасту, маніпуляціями з квітами. Але головне достоїнство алгоритму - швидкість роботи. Для порівняння хешів цього типу використовується функція нормованого відстані Хеммінга (докладніше в розділі 3.2.1).

3.1.2 Discrete Cosine Transform Based Hash

Дискретне косинусное перетворення (ДКП, DCT) [8] - одне з ортогональних перетворень, тісно пов'язане з дискретним перетворенням Фур'є (ДПФ), і є гомоморфизмом його векторного простору. ДКП, як і будь-який Фур'є-орієнтоване перетворення, висловлює функцію або сигнал (послідовність з кінцевого числа точок даних) у вигляді суми синусоїд з різними частотами і амплітудами. ДКП використовує тільки косинусні функції, на відміну від ДПФ, використовує і косинусні, і синусні функції.
Існує 8 типів ДКП [8]. Найпоширеніший - другий тип. його ми і будемо використовувати для побудови хеш-функції.
Визначення 3.1 (Другий тип ДКП):

Нехай $x[m]$, де $m = 0, \dots, N-1$ – послідовніть сигналу дліни N. Визначимо другий тип дискретного косинусного перетворення, як

$$X[n] = \sqrt{\frac{2}{N}} \sum_{m=0}^{N-1} x[m] \cos\left(\frac{(2m+1)n\pi}{2N}\right), n = 0, \dots, N-1$$

Цю формулу можна записати наступним чином:

$$X[n] = \sum_{m=0}^{N-1} c[n,m]x[m], n = 0, \dots, N-1$$

Де $c[n,m]$ елемент матриці ДКП на перетині рядка з номером n і стовпчика з номером m.

Визначення 3.2 (ДКП-матриця). ДКП-матриця визначається як:

$$c[n,m] = \sqrt{\frac{2}{N}} \sum_{m=0}^{N-1} \cos\left(\frac{(2m+1)n\pi}{2N}\right), \qquad m,n = 0, \dots, N-1$$

Рівняння 3.2 дуже зручно, коли ДКП обчислюється програмним шляхом. ДКП-матриця (3.3) може бути обчислена заздалегідь для будь-якої необхідної довжини. Таким чином, ДКП може бути представлено в вигляді:

$$\text{ДКП(I)} = \text{M} \cdot \text{I} \cdot \text{M}',$$

де M - ДКП-матриця, I - зображення квадратного розміру, збігається з розміром M, M '- зворотна матриця.

Різні властивості ДКП можуть бути використані для створення перцептивних хеш-функцій. Низькочастотні коефіцієнти ДКП найбільш стабільні до маніпуляцій з зображеннями. Це відбувається тому, що більша частина інформації сигналу, як правило,

зосереджена в декількох низькочастотних коефіцієнтах. це

властивість використовується, наприклад, при стисненні зображення за допомогою

стандарту JPEG. В якості матриці I зазвичай береться зображення,

спрощене (за допомогою різних фільтрів, наприклад, знебарвлення)

і стислий до розміру 32x32. В результаті виходить матриця ДКП (I), в лівому

верхньому кутку якої і знаходяться низькочастотні коефіцієнти. щоб

побудувати хеш береться лівий верхній блок частот 8x8. Потім з цього

блоку будується хеш розміром 8 байт за допомогою знаходження середнього і

побудови ланцюжка біт (також, як в Simple Based Hash).

Отже, перерахуємо кроки побудови хешу за допомогою даного алгоритму:

 Прибрати колір. Для придушення непотрібних високих частот;

Застосувати медіанний фільтр [9]. Для зменшення рівня шуму.

Суть медіанного фільтра в тому, що зображення розбивається на так

звані «вікна», потім кожне вікно замінюється медіаною [10] для

сусідніх вікон;

 Зменшити зображення до розміру 32x32;

Застосувати ДКП до зображення;

 Побудувати хеш.

Головні переваги такого хешу: стійкість до малих

поворотам, розмиття і стисненню зображення, а також швидкість порівняння

хешів, завдяки їхньому маленькому розміру. Для порівняння хешів цього типу використовується функція відстані Хеммінга (докладніше в розділі 3.2.1).

3.1.3 Radial Variance Based Hash

Ідея алгоритму Radial Variance Based Hash полягає в побудові променевого вектора дисперсії (ЛВД) на основі перетворення Радону [6]. Потім до ЛВД застосовується ДКП і обчислюється хеш.

Перетворення Радону - це інтегральне перетворення функції багатьох змінних вздовж прямої. Воно стійке до обробки зображень за допомогою різних маніпуляцій (наприклад, стиснення) і геометричних перетворень (наприклад, поворотів). У двовимірному випадку перетворення Радону для функції f (x, y) виглядає так:

$$R(s,\alpha) = \int_{-\infty}^{+\infty} f(scos\alpha - z\,sin\alpha, s\,sin\alpha + z\,cos\alpha)dz$$

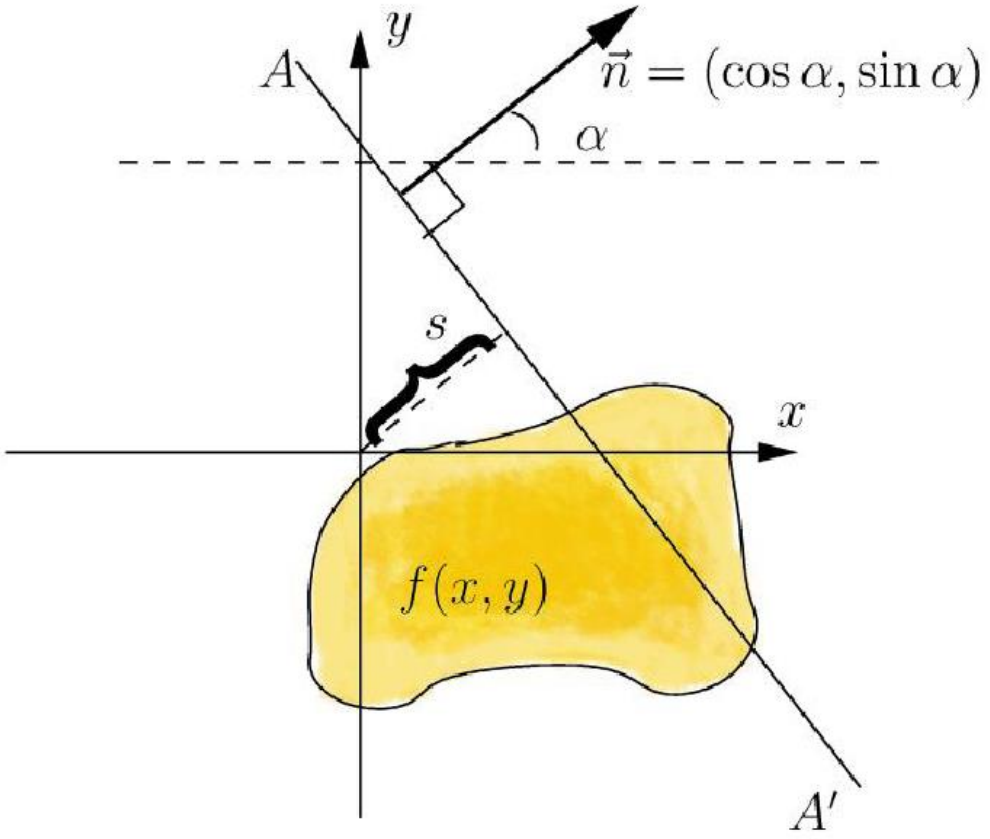
Перетворення Радону має простий геометричний зміст - це інтеграл від функції вздовж прямої, перпендикулярної вектору

$$\vec{n} = (\cos\alpha, \sin\alpha)$$

і що проходить на відстані s (виміряного уздовж вектора $\vec{n}$, з відповідним знаком) від початку координат. На рисунку $R(s,\alpha)$ це інтеграл від $f(x,y)$ уздовж прямої AA'.

Щоб перетворення Радону розширити для дискретних зображень, лінійний інтеграл по прямій

$$d - \frac{1}{2} \leq x\cos\alpha + y\sin\alpha \leq d + \frac{1}{2}$$



(Малюнок 3.2 Двовимірний перетворення Радону)

Пізніше було виявлено, що краще використовувати дисперсію замість суми значень пікселів уздовж лінії проекції [6]. дисперсія набагато краще обробляє яскравості розриви вздовж лінії проекції. такі яскравості розриви з'являються через країв, які ортогональні лінії проекції.

Тепер визначимо променевої вектор дисперсії. Нехай $\Gamma(\alpha)$ - набір пікселів на лінії проекції, яка відповідає цьому кутку. Нехай (x', y')

- координати центрального пікселя на зображенні. $(x, y) \in \Gamma(\alpha)$ тоді і тільки тоді, коли

$$-\frac{1}{2} \leq (x - x') \cos \alpha + (y - y') \sin \alpha \leq \frac{1}{2}$$

Означення 3.3. (Променевий вектор дисперсії)

Нехай $I(x, y)$ позначає яркість пікселя (x, y), $\#\Gamma(\alpha)$ потужність множини, тоді променевий вектор дисперсії $R[\alpha]$, де $\alpha = 0,1,2, \dots ,179$, визначимо як

$$R[\alpha] = \frac{\sum_{(x,y)\in\Gamma(\alpha)} I^2(x, y)}{\#\Gamma(\alpha)} - \left(\frac{\sum_{(x,y)\in\Gamma(\alpha)} I(x, y)}{\#\Gamma(\alpha)} \right)^2$$

Досить побудувати вектор по 180 значень кута, так як перетворення Радону є симетричним. отриманий вектор можна використовувати для побудови хешу, але в цьому алгоритмі пропонується ще деяке поліпшення: застосувати ДКП до отриманого вектору. В результаті вийде вектор, що успадковує всі важливі властивості ДКП. Як хешу беруться перші 40 коефіцієнтів отриманого вектора, які відповідають низьким частотам. Таким чином, розмір отриманого хешу - 40 байт.

Отже, перерахуємо кроки побудови хешу за допомогою даного алгоритму:

- Прибрати колір. Для придушення непотрібних високих частот;
- Розмити зображення (blurring) за допомогою використання Гауссова розмиття [11]. Зображення перетвориться за допомогою функції Гаусса. Для придушення деяких шумів;
- Застосувати гамма-корекцію. Для усунення блеклости зображення.

 Побудувати променевої вектор дисперсії;

Застосувати ДКП до вектору дисперсії;

 Побудувати хеш.

Для порівняння хешів цього типу використовується пошук піку взаімнокорреляціонной функції (докладніше в розділі 3.2.2).

3.1.4 Marr-Hildreth Operator Based Hash

Оператор Марра-Хілдрет [16] дозволяє визначати краю на зображенні. Взагалі кажучи, кордон на зображенні можна визначити як край або контур, що відокремлює сусідні частини зображення, які мають порівняно відмінні характеристики відповідно до деякими особливостями. Цими особливостями можуть бути колір або текстура, але частіше за все в якості них використовується сіра градація кольору зображення (яскравість).

Результатом визначення меж є карта кордонів. Карта кордонів описує класифікацію меж для кожного пікселя зображення.

Якщо кордону визначати як різка зміна яскравості, то для їх знаходження можна використовувати похідні або градієнт. нехай функція $f_c(x)$ позначає рівень яскравості для лінії (одновимірний масив пікселів). Перший підхід визначення меж полягає в знаходженні локальних екстремумів функції, тобто перших похідних. Другий підхід (метод Лапласа) полягає в знаходженні

друге похідних $f_c(x)$.

Обидва підходи можуть бути адаптовані для випадку двовимірних дискретних зображень, але з деякими проблемами. для знаходження похідних в дискретному випадку потрібно апроксимація. Крім того, шум на зображенні може значно погіршити процес пошуку кордонів. Тому перед визначенням меж до зображення потрібно застосувати будь-якої фільтр, що пригнічує шуми. для побудови хешу був обраний алгоритм, який використовує оператор Лапласа (2 підхід) і фільтр Гаусса.

Визначення 3.4 (Безперервний Лапласіан, оператор Лапласа):

Нехай $f_c(x, y)$ визначає яскравість на зображенні. тоді безперервний Лапласіан визначимо як:

$$\nabla^2 f_c(x, y) = \frac{\partial^2 f_c(x, y)}{\partial x^2} + \frac{\partial^2 f_c(x, y)}{\partial y^2}$$

Обернення у нуль $\nabla^2 f_c(x, y)$ це і є точки, що відповідають межі функції $f_c(x, y)$, так як це точки, при яких друга похідна наближається до нуля. Різні фільтри (дискретні оператори Лапласа) можуть бути отримані з безперервного Лапласіан. Такий фільтр, $h(n_1, n_2)$ може бути застосований до дискретного зображення за допомогою використання згортки функцій. Оператор Лапласа для зображення $f(n_1, n_2)$ можна переписати наступним чином:

$$\nabla^2 f(n_1, n_2) = f(n_1, n_2) * h(n_1, n_2)$$

де * позначає згортку функцій. Щоб побудувати карту меж, потрібно

знайти звернення в нуль дискретного оператора $\nabla^2 f(n_1, n_2)$.

Тепер розглянемо оператор Марра-Хілдрет. Він також називається Лапласіан від фільтра Гаусса (LoG) - це особливий тип дискретного оператора Лапласа. LoG конструюється за допомогою застосування оператора Лапласа до фільтру (функції) Гаусса. особливість цього оператора в тому, що він може виділяти кордону в певному масштабі. Змінну масштабу можна варіювати для того, щоб краще виявити кордону.

Визначення 3.5 (Фільтр Гаусса): Фільтр Гаусса визначимо як:

$$g_c(x, y) = e^{-\frac{x^2+y^2}{2\sigma^2}}$$

Згортку з операцією Лапласа можна поміняти місцями, тому що похідна і згортка - лінійні оператори:

$$\nabla^2[f_c(x, y) * g_c(x, y)] = [\nabla^2 g_c(x, y)] * f_c(x, y)$$

Ця властивість дозволяє раніше обчислити значення оператора $\nabla^2 g_c(x, y)$ оскільки він ніяк не залежить від зображення $f_c(x, y)$.

Визначення 3.6 (оператор Марра-Хілдрет, Лапласіан від фільтра Гаусса, LoG). LoG $h_c(x, y)$ визначимо як

$$h_c(x, y) = \nabla^2 g_c(x, y) = \frac{x^2 + y^2 - 2\sigma^2}{\sigma^4} e^{-\frac{x^2+y^2}{2\sigma^2}}$$

Щоб використовувати LoG в дискретної формі, зробимо дискретизацию даного рівняння, підставивши потрібну змінну масштабу. за замовчуванням її значення приймається як 1.0. Потім фільтр можна застосувати до зображення, використовуючи дискретну згортку.

Визначення 3.7 (Згортка з бібліотеки CImg):

Нехай x, y, z - піксельна ширина, довжина і глибина зображення I.

Результат R згортки зображення I с маскою M визначимо як:

$$R(x,y,z) = \sum_{i,j,k} I(x-i, y-j, z-k)M(i,j,k)$$

Тепер перерахуємо кроки алгоритму, що використовує для побудови хешу оператор Марра-Хілдрет:

 Прибрати колір. Для придушення непотрібних високих частот;

 Провести зображення в розмір 128x128;

 Розмити зображення (blurring) за допомогою використання Гауссова розмиття. [11] Зображення перетвориться за допомогою функції Гаусса. Для придушення деяких шумів;

 Побудувати оператор Марра-Хілдрет;

Застосувати дискретну згортку до LoG і зображенню. вийде зображення, на якому чітко видно скачки яскравості.

 Перетворити зображення в гістограму. зображення розбивається на маленькі блоки (5x5), в яких підсумовуються значення яркостей.

a) Гаусове Розумієте b) Застосування LoG c) гістограма яркостей

Малюнок 3.4 Приклад застосування алгоритму Marr-Hildreth Operator Based Hash до зображення на малюнку 3.1

 Побудувати хеш з гістограми. Гістограма розбивається на блоки 3x3. Для цих блоків вважається середнє значення яскравості і використовується метод побудови ланцюжка бітів. виходить бінарний хеш розміром 64 байта.

Розмір отриманого хешу не маленький, проте, порівняння двох хешів займає досить незначний час в порівнянні з Radial алгоритмом, так як використовується функція нормованого відстані Хеммінга (докладніше в розділі 3.2.1). Також такий алгоритм чутливий до поворотів зображення, але стійкий до масштабування, затемнення, стиску. максимально хороші результати алгоритм показує, якщо спам-фрагмент сильно відрізняється в яскравості від решти картинки (наприклад, напис білого кольору).

3.2 Функції порівняння перцептивних хеш-значень

Перцептивні хеш-функції обчислюють схожі хеш-значення для схожих зображень. Існує кілька способів визначити наскільки схожі хеш-значення. В 4 описаних в попередньому розділі алгоритмах використовується дві функції порівняння: відстань Хеммінга (Hamming distance) і пік взаімнокорреляціонной функції (Peak of Cross Correlation, PCC). Нижче розглянемо їх докладніше.

3.2.1 Відстань Хеммінга

Відстань Хеммінга визначає кількість різних позицій між двома бінарними послідовностями.

Визначення 3.8 (Відстань Хеммінга)

Нехай A - алфавіт кінцевої довжини.

$$x = (x_1, \dots, x_n), y = (y_1, \dots, y_n)$$

бінарні послідовності (вектори). Відстань Хеммінга Δ між x і y визначимо як:

$$\Delta(x, y) = \sum_{x_i \neq y_i} 1, \quad i = 1, \dots, n$$

Цей спосіб порівняння хеш-значень використовується в методі DCT Based Hash. Хеш займає розмір 8 байт, тому відстань Хеммінга лежить в відрізку [0, 64] Чим менше значення Δ, тим більше схожі зображення.

Визначення 3.9 (Нормоване відстань Хеммінга)

Для полегшення порівняння відстань Хеммінга можна нормувати з

допомогою довжини векторів:

$$\Delta_{\mathrm{n}}(x,y) = \frac{1}{n}\sum_{x_i \neq y_i} 1, \quad i = 1, \dots, n$$

Нормоване відстань Хеммінга використовується в алгоритмах Simple Hash та Marr-Hildreth Operator Based Hash. Відстань Хеммінга лежить в проміжку [0,1] і чим ближче Δ_{n} до 0, тим більше схожі зображення.

3.2.2 Пік взаімнокорреляціонной функції

Визначення 3.10 (Кореляційна функція) Кореляцію між двома сигналами визначимо як:

$$r_{xy}(T) = \int_{-\infty}^{+\infty} x(t)y(t+T)dt$$

Де $x(t), y(t)$ – дві неперервні дійсні функції. Функція $r_{xy}(T)$ описує зсув цих двох сигналів щодо часу T. Змінна T визначає наскільки сигнал зміщений ліворуч.

Якщо сигнали $x(t), y(t)$ функція $r_{xy}(T)$ називається взаємокореляційною.

Визначення 3.11 (Нормована взаємокореляційна функція)

Нехай $x_i,\ y_i, i = 0, \dots, N-1$ дві послідовності дійсних чисел, а N – довжина обох послідовностей. Нормовану взаємокореляційну функцію с затримкою d визначимо як

$$r_d = \frac{\sum_i (x_i - mx)(y_{i-d} - my)}{\sqrt{\sum_i (x_i - mx)^2}\sqrt{\sum_i (y_i - my)^2}}$$

де mx і my позначають середнє значення для відповідної послідовності.

Пік взаємокореляційної функції (PCC) - максимальне значення функції r_d яке може бути досягнуто на проміжку $d = [0, N]$.

PCC використовується для порівняння хеш-значень в алгоритмі Radial Variance Based Hash. PCC $\in [0,1]$, чим більше його значення, тим більше схожі зображення.

особливості реалізації

4.1 Засоби реалізації

Для виконання поставлених завдань було необхідно адаптувати описані в розділі 3 алгоритми для завдання пошуку спам-зображень. Для реалізації алгоритму Simple Hash була обрана бібліотека з відкритим вихідним кодом Open Computer Vision [14]. Ця бібліотека, реалізована на мовах C / C ++, надає набір типів даних і алгоритмів для обробки зображень методами комп'ютерного зору. Реалізація інших трьох алгоритмів була взята з бібліотеки pHash [13]. Це бібліотека з відкритим вихідним кодом, написана на C / C ++, реалізує кілька перцептивних хеш-алгоритмів для зображень, тексту, відео та аудіо. Алгоритми для зображень в pHash реалізовані за допомогою використання бібліотеки CImg [15]. ця бібліотека, написана на C ++, надає класи і функції обробки зображень.

4.2 Пристрій спам-фільтра

Отже, яким чином фільтр спам-зображень повинен працювати? Припустимо, що є сервіс, схильний до спам-атакам. Модератор спаму цього сервісу при виявленні спам-зображення перевіряє, є Чи зображення зі схожим спам-фрагментом в базі даних і в разі знаходження таких - видалити їх. Завдання спам-фільтра полегшити роботу модератора і провести пошук зображень автоматично.

На вхід фільтру подається безліч паттернів. безліч

патернів вдає із себе набір пар: шлях до спам-зображенню і

відносні координати спам-фрагмента на цьому зображенні:

typedef pair <const char *, Coord *> Pattern;

set <Pattern> setOfPatterns;

де Coord – структура з чотирма координатами $x1, x2, y1, y2 \in [0, 1]$.

Суть множини патернів полягає в тому, що фільтр будує хеш

значення для всіх спам-зображень серед патернів і при пошуку спаму

порівнює хеші серед патернів з хешем перевіряється зображення.

У разі, якщо хеш-кодування досить близькі (докладніше в наступному розділі)

хоча б для одного патерну, фільтр оголошує відповідне

зображення спамом.

Розглянемо сигнатуру методів для роботи з безліччю патернів:

void insert (Pattern pattern, int alg);

void remove (Pattern pattern);

де set - контейнер з бібліотеки STL. Метод insert () бере один з 4

алгоритмів з номером alg і за допомогою нього визначає, чи є схожі

зображення в безлічі патернів. Якщо схожих зображень немає, то

зображення додається до безлічі патернів.

executeAlgorithm (int alg) - метод, що виконує конкретний

перцептивний алгоритм.

Сигнатура самого фільтра виглядає так:

void searchSpam (set <Pattern> setOfPatterns,

list <uint8_t> images, int alg);

де images - список зображень, серед яких шукається спам з допомогою алгоритму alg.

В результаті роботи спам-фільтра зображення діляться на 3 групи:

1) «точно спам»

2) «можливо спам». Коли перцептивний алгоритм порівнює два хешу, на виході повертається «схожість» - відстань між хешамі, отримане за допомогою деяких метрик (розділ 3.2). Для кожного алгоритму існує діапазон «подібностей», в який потрапляє як спам, так і не спам. З цієї причини була введена дана група.

3) «точно не спам».

4.3 Адаптація алгоритмів

У розділі 3 були описані 2 типу алгоритмів:

Алгоритми, в яких хеш будується з матриці частот (DCT Based Hash та Radial Variance Based Hash). Кожен елемент в матриці частот (і відповідно в хеше) залежить від усіх пікселів на зображенні.

 Алгоритми, в яких хеш будується з спрощеного зображення або гістограми яркостей (Simple Hash, Marr-Hildreth Operator Based Hash). Кожен коефіцієнт в отриманому хеше залежить від декількох сусідніх пікселів на зображенні.

Алгоритми 1 типу не підходять для завдання пошуку спам-фрагмента при умови, що його місце на зображенні не фіксоване: в отриманих хеш-значних зображень неможливо провести пошук хеш-значень фрагмента. Алгоритми 2 типу для цього завдання підходять.

Розглянемо змінені сигнатури методів, що реалізують алгоритми 1 типу (з бібліотеки pHash):

int ph_dct_imagehash (uint8_t image, ulong64 & hash,
Coord * coord); // DCT Based Hash

int ph_image_digest (const char * file, double
sigma, double gamma, Digest & digest, Coord * coord);
// Radial Variance Based Hash

де image - зображення;

hash, digest - змінні, в які буде записаний хеш після завершення роботи методів;

sigma, gamma - змінні, що відповідають за розмиття і гамма корекцію;

coord - відносні координати спам-фрагмента;

Обидва методи спочатку обрізають зображення по координатам coord, а вже потім виконують всі інші кроки алгоритмів і будують хеш НЕ для всього зображення, а для спам-фрагмента.

Розглянемо сигнатури методів, що реалізують алгоритми 2 типу:

uint8_t * ph_mh_imagehash (uint8_t image, int

& N, float alpha, float lvl, Coord * coord); // Marr-

Hildreth Operator Based Hash, з pHash

__int64 * calcImageHash (uint8_t image, Coord

* Coord); // реалізований з використанням OCV

де image - зображення;

alpha, lvl - змінні, що відповідають за масштабування і його рівень;

N - розмір хешу;

coord - відносні координати спам-фрагмента;

Обидва методи також можуть вважати хеш тільки спам-фрагмента. В

випадку, якщо становище фрагмента невідомо, змінна coord = null, і

вважається хеш всього зображення. У цьому випадку функція розрахунку

подібності хеш-значень (для обох алгоритмів - нормоване

відстань Хеммінга) працює по-іншому.

Так як хеш будується з верхнього лівого кута зображення направо і

вниз, то є відповідність між коефіцієнтами хешу і пікселями

зображення.

Визначення 4.1 (Нормоване відстань Хеммінга для спам-

фрагмента) Нехай *Coord(hash, size)* – множина всіх можливих розташувань фрагменту розміром *size=(height, width)* для хеша *hash*.

$height, width \in [0,1]$

Sub(hash,coord) функція, яка повертає

«Подхеш» для хешу hash, відповідний фрагменту з відносними

координатами coord. Тоді нормоване відстань Хеммінга для фрагменту розміром *size* між хешами A та B визначимо як

$$\Delta_n^{size}(A,B) = \min_{c \in Coord\,(A,size)} (\Delta_n(Sub(A,c),B)$$

Як хешу A виступає хеш зображення, що перевіряється на

наявність спам-фрагмента, а в якості B - «подхеш», відповідний

спам-фрагменту зображення-підкладки з безлічі патернів.

Розглянемо сигнатуру методу нормованого відстані Хеммінга

для фрагмента:

double hammingdistanceFragment (__int64

* CheckedHash, int checkedLen, __int64 * fragmentHash,

int fragmentLen, Size size, int width, int height);

де checkedHash - перевіряється хеш;

fragmentHash - хеш спам-фрагмента;

checkedLen, fragmentLen - довжини хешів;

width - розміри в бітах рядки в хеше, що відповідає одній рядку на

зображенні;

height - кількість рядків в хеше;

Size - структура, що зберігає розміри спам-фрагмента.

Кроки виконання методу наступні:

 побудова безлічі Coord(checkedHash, size)

побудова всіх можливих нормованих відстаней Хеммінга;

 пошук мінімального з них.

експерименти

Для тестування і аналізу результатів роботи розробленого спам-фільтра була обрана база даних з 60000 зображень, з яких 700 є спамом. У безліч паттернів додавалися зображення 3 типів спаму (рис. 5.1)

a) фрагмент «kibergrad.com»

b) фрагмент «best-mp3.ru»

c) фрагмент «o5wap.ru»

Малюнок 5.1 Приклади зображень-підкладок з різними фрагментами спаму

5.1 Аналіз результатів роботи алгоритмів

Розіб'ємо всі зображення на 2 класу: спам (1 клас) і не спам (2 клас). Існує 5 різних результатів роботи фільтра (Таблиця 5.1).

Результат роботи	1 клас	2 клас
Позитивне	Істинно-позитивний	Хибно-позитивний

спрацьовування		(Помилка 2 роду)
Негативне спрацьовування	Хибно-негативний (Помилка 1 роду)	Істинно-негативний
Невизначене спрацьовування	Група «Можливо спам», в якій знаходяться	

Таблиця 5.1 Матриця результатів

Для того, щоб оцінити якість отриманих результатів, використовувалися наступні метрики (рис. 5.2):

 Коефіцієнт помилкового прийняття (False Accept Rate, FAR) .FAR - кількість всіх помилково-позитивних спрацьовувань (помилок 2 роду). Хибно-позитивний спрацьовування відбувається в разі, якщо фільтр визначає зображення як спам, яке насправді не спам.

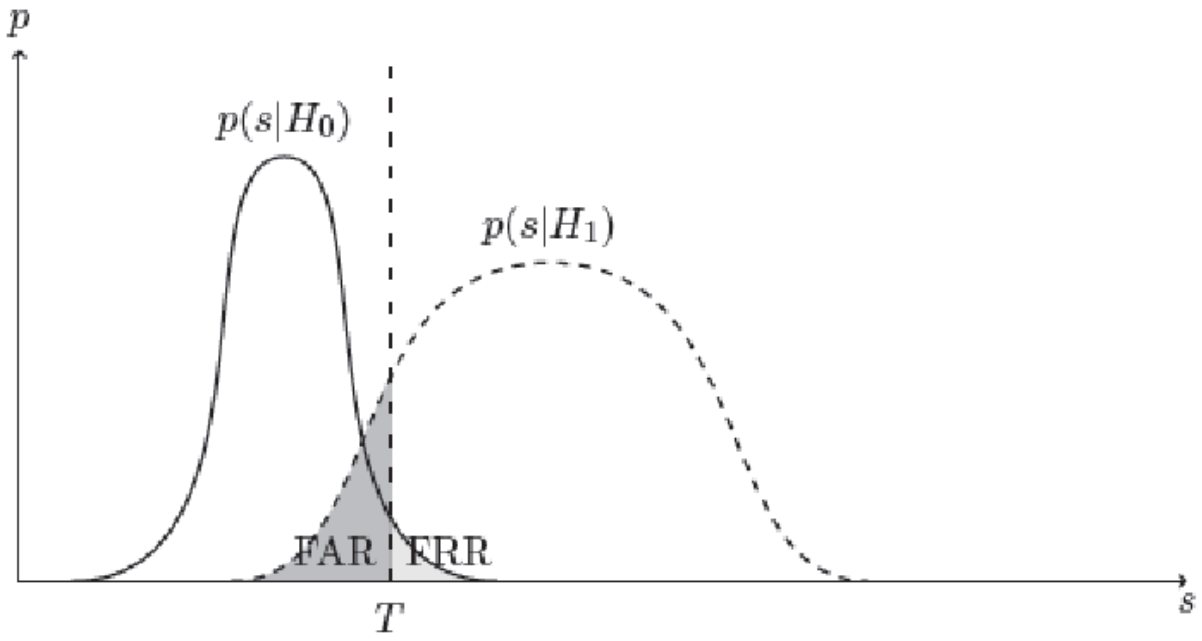
 Коефіцієнт помилкового відхилення (False Reject Rate, FRR) - кількість всіх хибно-негативних спрацьовувань (помилок 1 роду). Хибно-негативний спрацьовування відбувається в разі, якщо фільтр визначає зображення як не спам, яке насправді спам.

Результатом порівняння двох хешів є «схожість» s. якщо s більше, ніж деякий вибраний поріг T, зображення вважаються перцептивно схожими. Завдяки варьированию порога можливо збільшувати кількість помилок 1 роду і зменшувати кількість помилок 2 роду і навпаки. Крім того, була введена додаткова група «Можливо спам». У неї потрапляють зображення зі подібностями S такими, Що

$$T_1 < S < T_2$$

Де T_1, T_2 такі пороги, що $T_1 < T < T_2$.

Група «Можливо спам» дозволяє істотно зменшити кількість помилок 1 і 2 роду.



На малюнку 5.1 T - поріг, S - схожість двох хешів, $p(s|H_0)$ – ймовірність того, що при подібності s зображення - насправді спам, $p(s|H_1)$ - ймовірність того, що при подібності s зображення - не спам.

Отже, в безліч паттернів були поміщені 5 зображень 3 типів спаму (рис. 5.1), 3 з яких представляли різні варіації фрагмента «Kibergrad» для кращого виявлення такого спаму. В ході виконання експериментів були виявлені найкращі пороги подібності для кожного алгоритму, що мінімізують помилково-позитивні і хибно-негативні спрацьовування. Потужність групи «можливо спам» було вирішено обмежити 1% від потужності БД.

Результати представлені на таблиці 5.2. У 1 і 2 рядку в процентах вказана частина спам-зображень від загальної їх кількості, що потрапила в данну групу. У рядку «можливо спам» зазначено кількість потрапили в цю групу зображень і кількість опинилися їх них спамом після ручної перевірки. У рядку для хибно-позитивного спрацьовування вказана ймовірність помилки у відсотках щодо все БД.

Срабатывание\Алгоритм	SH	DCT	RV	MH
Истинно-позитивное	71%(500)	79%(545)	82%(565)	75%(530)
«Возможно спам» 3% (25 из 350)		11%(75 из 280)	17%(127 из 321)	
Ложно-позитивное	0%	0,7%(5)	0%	1%(7)
Ложно-негативное	26%	10%	1%	17%

Таблиця 5.2 Результати роботи алгоритмів.

Таким чином, помилково-позитивних спрацьовувань вдалося уникнути практично для всіх алгоритмів, а кращі результати показав алгоритм Radial Variance Based Hash - 82% як «точно спам» і 17% «можливо спам ». Завдяки групі «можливо спам» вийшло мінімізувати кількість хибно-негативних спрацьовувань для даного алгоритму.

5.2 Аналіз швидкості алгоритмів

Швидкість роботи кожного з 4 алгоритмів була перевірена при їх виконанні для всіх 60000 зображень з БД.

Характеристика	SH	DCT		RV
Загальний час обробки	БД(мин.)	7	50	66
Среднее время обработки 1 изображения(мс.)		6	47	63
Среднее время сравнения 100 хешей(мс.)	<<1	<<1	50	1
Среднее время поиска подстроки в 1 хеше(мс.)	5	—	—	3

Таблиця 5.3 Результати перевірки швидкості роботи перцептивних хеш алгоритмів.

Результати показали, що Simple Hash є найшвидшим алгоритмом, серйозно випереджаючи інші алгоритми. для алгоритмів SH, DCT і MH функції порівняння значень хешів (відстань Хеммінга) працюють на кілька порядків швидше в порівнянні з обчисленням хешу, тому в цих випадках можна не турбуватися про потужності безлічі патернів, якщо не стоїть завдання пошуку підрядка в хеше. Для алгоритму Radial функція порівняння (кореляція) працює значно повільніше: порівняння ~ 150 хешів еквівалентно обчисленню 1 хешу.

Середній час пошуку підрядка в 1 хеше істотно: для

алгоритму SH - 5мс., для MH - 3мс. для одного хешу. Кращий результат для MH можна пояснити тим, що розмір хешу у нього 64 байта, в той час як у SH - 128. Для пошуку підрядка в хеше використовувався «Наївний» алгоритм перебору всіх можливих фрагментів (визначення 4.1 розділу 4.3). Алгоритми DCT і RV не підтримують пошук підрядка (докладніше в розділі 4.3), тому в таблиці для них варто прочерк. Висновок: завдання пошуку підрядка необхідно вирішувати при маленької потужності безлічі патернів.

висновок

В ході виконання даної дипломної роботи були вивчені різні існуючі підходи для обчислення перцептивних хешів зображень. Було адаптовано 4 перцептивних хеш-алгоритму для завдання пошуку спам-зображень. В ході ряду експериментів були визначені алгоритми, що дають найбільш якісні результати розпізнавання спаму. Було створено API для зручного використання алгоритмів.

В майбутньому планується введення алгоритмів в експлуатацію і розробка користувальницького інтерфейсу для модераторів спаму в соціальної мережі

Література.

[1] Статья по теме «Доля мусора в почтовом трафике».

http://www.computery.ru/news/news2010.php?nid=8302

[2] Egmont-Petersen, M., de Ridder, D., Handels, H. "Image processing with neural networks - a review". Pattern Recognition 35 (10), pp. 2279–2301(2002)

[3] Christoph Zauner. Implementation and Benchmarking of Perceptual Image Hash Functions (2010)

[4] Locality-sensitive hashing.

http://en.wikipedia.org/wiki/Locality-sensitive_hashing

[5] Simple and DCT perceptual hash-algorithms.

http://www.hackerfactor.com/blog/index.php?/archives/432-Looks-Like-It.html

[6] Standaert, F.X., Lefebvre, F., Rouvroy, G., Macq, B.M., Quisquater, J.J., and Legat, J.D.: Practical evaluation of a radial soft hash algorithm. In Proceedings of the International Symposium on Information Technology: Coding and Computing (ITCC), vol. 2, pp. 89-94. IEEE, Apr. 2005

[7] D. Marrand, E. Hildret. Theory of edge detection, pp. 187-215 (1979)

[8] http://en.wikipedia.org/wiki/Discrete_cosine_transform

[9] http://en.wikipedia.org/wiki/Median_filter

[10] http://en.wikipedia.org/wiki/Median

[11] http://en.wikipedia.org/wiki/Gaussian_blur

[12] Zeng Jie. A Novel Block-DCT and PCA Based Image Perceptual Hashing

Algorithm. IJCSI International Journal of Computer Science Issues, Vol. 10, Issue 1, No 3, January 2013

[13] Perceptual hash library pHash. http://www.phash.org/

[14] Open Computer Vision library. http://opencv.org/

[15]The CImg library. http://cimg.sourceforge.net/

[16] http://de.wikipedia.org/wiki/Marr-Hildreth-Operator