

# МЕТОД АРХІВУЮЧОЇ КЛАСТЕРИЗАЦІЇ ІНФОРМАЦІЙНИХ ПОВІДОМЛЕНЬ

О. О. Сірий<sup>1</sup>

<sup>1</sup> Національний технічний університет України "Київський політехнічний інститут"

## Анотація

В даній статті представлено метод визначення характеристик текстів та їх класифікації за допомогою архівування. Використовуючи прямий зв'язок архівування за допомогою алгоритмів LZ77 і Хаффмана з ентропією, виділяються ознаки тексту, що дозволяють визначати мову його написання, стиль, авторство, кластеризувати масиви даних за їх належністю до певної тематики.

**Ключові слова:** архівація, ентропія, розпізнавання тексту, спам, фішинг, LZ77, алгоритм Хаффмана.

## Вступ

Важливе місце в спілкуванні за допомогою мережі Інтернет займає обмін текстовими повідомленнями. Спілкування з колегами, друзями, родиною, отримання інформації про актуальні події навколишнього світу, ознайомлення з новинами - все це представлено текстовою інформацією у повідомленнях електронної пошти, меседжах соціальних мереж, постах блогів і сайтів новин. Таке широке використання даного виду передачі інформації безсумнівно притягує в цю сферу тих, хто не завжди слідує правилам суспільства. Спам, фішинг, відмова від авторства, дезінформація — все це і багато іншого використовується шахраями у власних цілях. Виходячи з цього, постає проблема аналізу повідомлень на рівні їх змісту, стилю написання. Повний вербальний аналіз текстів з урахуванням рівня технологій, незважаючи на все, залишається важкою задачею. Тому для цього можуть бути використані статистичні методи аналізу. Існують методи, що базуються на математичних властивостях інформації (зокрема, ентропії), що дозволяють виділяти спільні ознаки в близьких за походженням або авторством текстових повідомленнях.

## 1. Постановка задачі

Проблема оптимального кодування тексту, зображення або будь-якого іншого виду інформації активно вивчалася в минулому столітті. Зокрема, американський науковець Клод Шенон виявив, що існує обмеження на можливість кодування послідовності. Це обмеження — це ентропія послідовності. Є багато еквівалентних визначень ентропії, але, ймовірно, найкраще визначення в даному контексті — це ентропія Хайтіна-Колмогорова: ентропія послідовності символів — це довжини (в бітах) найменшої програми, яка на виході дає цю послідовність. Це визначення є абстрактним. Зокрема, неможливо навіть теоретично знайти згадану програму. Проте є такі алгоритми, які найбільше наближаються до цієї теоретичної ме-

жі. Це компресори файлів або архіватори. Архіватор бере файл і намагається перетворити його в настільки короткий відповідник, наскільки це можливо без втрати змісту.

## 2. Алгоритм LZ77

Одним з перших алгоритмів стиснення без втрат є алгоритм Лемпеля-Зіва (LZ77). LZ77 використовує вже переглянуту частину повідомлення в якості словника. Щоб досягти стиснення, алгоритм намагається замінити наступний фрагмент повідомлення на вказівник у словнику.

У якості моделі даних LZ77 використовує «ковзаюче» по повідомленню вікно, розділене на дві нерівні частини. Перша частина більша за розміром і включає в себе уже переглянуту частину повідомлення. Друга, значно менша, — буфер, що містить у собі ще не закодовані символи. Як правило, розмір вікна — кілька кілобайт. Буфер значно менший, зазвичай не більше ста байтів. Алгоритм намагається знайти фрагмент в словнику, який збігається зі змістом буфера.

Алгоритм LZ77 видає коди, що складаються з 3 елементів:

- 1) Зміщення в словнику відносно його початку підстроки, котра відповідає змісту буфера;
- 2) Довжина строки;
- 3) Перший символ в буфері після підстроки.

Після цього алгоритм переміщає вікно на довжину співпадаючої підстроки + 1 символ вліво. Якщо збіг не знайдено, алгоритм видає код  $<0, 0$ , перший символ в буфері  $>$  і продовжує свою роботу.

## 3. Алгоритм Хаффмана

Ще одним алгоритмом стиснення без втрат є алгоритм Хаффмана. Ідея алгоритму полягає в наступному: знаючи імовірність символів в повідомленні, можна описати процедуру побудови кодів змінної довжини, що складаються з цілої кількості бітів. Символи з більшою імовірністю відповідають більш

коротким кодам і навпаки. Коди Хаффмана володіють властивістю префіксності, що дозволяє декодувати їх однозначно.

Алгоритм побудови дерева кодування Хаффмана:

- 1) Символи вхідного алфавіту формують список вільних вузлів. Кожен лист має вагу, що відповідає імовірності його появи або кількості його входжень в текст;
- 2) Обираються два вільних вузли з найменшою вагою;
- 3) Створюється їх батьківський вузол, вага якого рівна їх сумарній вазі;
- 4) Батьківський вузол додається в список вільних вузлів, а його потомки видаляються з нього;
- 5) Одній дузі, що виходить з батьківського вузла ставиться у відповідність 0, іншій 1;
- 6) Кроки, починаючи з другого, повторюються до тих пір поки в списку вільних вузлів не залишиться тільки один вузол. Він буде вважатись коренем дерева.

Щоб визначити код для кожного з символів, що входять до повідомлення необхідно пройти шлях від листа дерева, що відповідає поточному символу, до його кореня, накопичуючи біти при переміщенні по гілкам дерева (перша гілка відповідає меншому біту). Отримана таким чином послідовність бітів буде кодом поточного символу, записаним в зворотному порядку.

Так як жоден з отриманих кодів не є префіксом попереднього, вони можуть бути однозначно декодовані при зчитуванні з потоку.

Класичний алгоритм Хаффмана має ряд суттєвих недоліків. По-перше, для відновлення змісту повідомлення декодер повинен знати таблицю частот, яку використовував кодер. Крім того, необхідно мати повну частотну статистику перед кодуванням, через що необхідно два проходи по тексту. По-друге, надлишковість кодування перетворюється в нуль лише тоді, коли імовірності кодованих символів є зворотними степенями двійки. По-третє, для джерела з ентропією, що не перевищує 1, застосування коду Хаффмана не має жодного сенсу.

Для спрощення виконання алгоритму була розроблена його модифікація — адаптивне стиснення. Воно дозволяє не передавати модель повідомлення разом з ним, і для нього необхідний лише один прохід по повідомленню як для кодування, так і для декодування.

При реалізації алгоритму адаптивного кодування Хаффмана найбільшою складністю є розробка процедури оновлення дерева кожним наступним символом. Існує метод модифікації уже існуючого дерева, так щоб відобразити обробку нового символу. Оновлення дерева при зчитуванні наступного символу складається з двох операцій.

Перша — збільшення ваги вузла дерева. Спочатку збільшуємо вагу листа, що відповідає поточному символу на одиницю. Потім збільшуємо вагу батьківського вузла, щоб привести його у відповідність до нового значення ваги нащадку. Цей процес продовжується поки ми не дістанемось кореня дерева. Серед-

нє число операцій збільшення ваги рівно середній кількості бітів, що необхідні для того, щоб закодувати символ.

Друга операція — перестановка вузлів дерева — необхідна у випадку коли збільшення ваги вузла приводить до порушення властивості впорядкованості дерева. Якщо і далі продовжувати обробляти збільшення ваги вузла, рухаючись до кореня, то дерево перестане бути деревом Хаффмана.

Щоб зберегти впорядкованість дерева кодування, алгоритм працює наступним чином. Нехай нова збільшена вага вузла рівна  $W + 1$ . Тоді починаємо рухатись по списку в сторону збільшення ваги, поки не знайдемо останній вузол з вагою  $W$ . Переставимо поточний вузол і знайдений між собою в списку, відновлюючи таким чином порядок в дереві (при цьому батьківські вузли кожного з вузлів теж зміняться). На цьому операція перестановки завершується.

Після перестановки операція збільшення ваги вузлів продовжується далі. Наступний вузол, вага якого буде збільшена алгоритмом — це новий батьківський вузол вузла, вага котрого викликала перестановку.

#### 4. Використання алгоритмів стиснення для визначення ентропії

Алгоритми стиснення забезпечують потужний інструмент для вимірювання ентропії. Перший висновок, який можна зробити, — це можливість вимірювання ентропії методом простої архівації тексту. У цій роботі ми використовуємо такий алгоритм для визначення віддаленості між парами текстів.

Найпростіший спосіб зрозуміти, звідки походить наше визначення, згадати поняття відносної ентропії, сутність якого легко зрозуміти на наступному прикладі.

Розглянемо два ергодичних джерела  $\mathcal{A}$  і  $\mathcal{B}$ , що генерують послідовності 0 і 1:  $\mathcal{A}$  генерує 0 з імовірністю  $p$  і 1 з імовірністю  $1 - p$ , в той час як  $\mathcal{B}$  генерує 0 з імовірністю  $q$  і 1 з імовірністю  $1 - q$ . Алгоритм стиснення при застосуванні до послідовності, згенерованої джерелом  $\mathcal{A}$ , зможе закодувати послідовність майже оптимально, тобто 0 буде кодований  $-\log_2(p)$  бітами і 1  $-\log_2(1 - p)$  бітами. Це оптимальне кодування не буде оптимальним для послідовності  $\mathcal{B}$ . Зокрема, ентропія на символ послідовності  $\mathcal{B}$  в кодуванні, оптимальному для  $\mathcal{A}$ , буде рівна  $-q \log_2(p) - (1 - q) \log_2(1 - p)$ , в той час як ентропія на символ послідовності  $\mathcal{B}$  в її оптимальному кодуванні буде дорівнювати  $-q \log_2(q) - (1 - q) \log_2(1 - q)$ . Кількість бітів на символ витрачених, щоб закодувати послідовність  $\mathcal{B}$  оптимальним кодуванням для  $\mathcal{A}$ , буде відносною ентропією  $\mathcal{A}$  і  $\mathcal{B}$ .

$$S_{AB} = -q \log_2\left(\frac{p}{q}\right) - (1 - q) \log_2\left(\frac{1 - p}{1 - q}\right)$$

Існує кілька способів вимірювання відносної ентропії. Однією з можливостей є, звичайно, використати приклад, описаний вище: використання оптимального кодування для даного джерела для кодування повідомлення іншого джерела. Ми використовуємо

схожий алгоритм. Для того, щоб визначити відносну ентропію між двома джерелами  $\mathcal{A}$  і  $\mathcal{B}$ , ми генеруємо довгу послідовність  $A$  джерелом  $\mathcal{A}$ , довгу послідовність  $B$ , а також коротку послідовність  $b$  джерелом  $\mathcal{B}$ . Створюємо нову послідовність  $A+b$  простим додаванням  $b$  після  $A$ . Після того архівуємо її, наприклад, за допомогою GZIP. Мірою довжини  $b$  в оптимальному кодуванні для  $A$  буде  $\Delta_{Ab} = L_{A+b} - L_A$ , де  $L_X$  – довжина в бітах архівованого файлу  $X$ . Відносна ентропія на символ  $S_A$  між  $\mathcal{A}$  і  $\mathcal{B}$  дорівнює

$$S_{AB} = \frac{(\Delta_{Ab} - \Delta_{Bb})}{|b|}$$

де  $|b|$  – кількість символів в послідовності  $b$  і  $\Delta_{Bb} = (L_{B+b} - L_B)/|b|$  – оцінка ентропії джерела  $\mathcal{B}$ .

## 5. Методика кластеризації

Даний метод може бути використаний для визначення мови написання тексту, його стилю, належності певному автору, кластеризації масивів даних за їх належністю до певної тематики. Основний підхід включає в себе такі етапи:

- 1) Збір банку даних для поставленої задачі;
- 2) Порівняння цільового тексту з кожним елементом банку;
- 3) Аналіз результатів.

Розглянемо це на прикладі. Нехай ми маємо текст  $X$ , що написаний невідомою для нас мовою. В даному випадку банком даних буде виступати бібліотека текстів, написаних різними мовами, та їхні відповідники в архівованому вигляді. Порівняння текстів проходить за наступним алгоритмом:

- 1) Частина тексту  $X$  додається до оригінального тексту  $A_i$  з бібліотеки (кожного разу використовується той самий уривок тексту  $X$ ).
- 2) Проводиться архівування, методом аналогічним до того, що був застосований для архівування елементів бібліотеки.
- 3) Обраховується різниця між довжиною архівованого тексту  $A_i$  та тексту отриманого в попередньому кроці.

Процедура повторюється для кожного тексту в бібліотеці. Найменше значення різниці буде свідчити про найбільше наближення мови даного тексту до мови тексту з бібліотеки.

Розглянемо чому попереднє твердження є вірним. Будемо використовувати алгоритм Хаффмана і вважатимемо, що тексти  $X$  та  $A_i$  в даному прикладі написані різними мовами. Коли архіватор розпочинає кодувати файл він починає процес формування дерева кодів. З часом зміни в дереві відбуваються все рідше, за рахунок того, що всі символи мови вже записані в нього і їх розподіл є усталеним. Досягнувши кінця тексту  $A_i$  ми отримуємо дерево кодів Хаффмана налаштоване на символи і закономірності їх слідування для певної мови. Коли архіватор починає опрацьовувати текст іншої мови з'являються нові символи або змінюється їх статистичний розподіл. У випадку зміни набору символів у дереві починають з'являтися нові листи. У порівняння з листами початкового набору символів їх вага буде

значно меншою, а отже і довжина коду для такого символу буде більшою. Нові, більші коди вплинуть на розмір архівованого тексту більшою мірою, ніж якби були використані ті самі коди, що вже є в дереві. У випадку мови зі схожим набором символів, вплив нових символів не буде таким значним, хоча він все рівно буде залишатись. У цьому випадку на темпи збільшення розміру файлу буде впливати розподіл символів. Коди Хаффмана вже не будуть оптимальними для цієї частини тексту, тому розмір шифрованого тексту буде не мінімально можливим.

Використання описаного методу для підтвердження належності повідомлення  $X$  певному автору має певні відмінності в підході. Для цієї ситуації банком даних буде виступати архів повідомлень користувача, що нас цікавить. Важливою умовою є те, що всі повідомлення повинні бути написані однаковою мовою, адже враховуючи попередній приклад — нові символи значною мірою впливають на процес. Розмір повідомлень не повинен бути однаковим для всіх елементів банку. Алгоритм буде виглядати таким чином:

- 1) З  $N$  повідомлень архіву генеруються (проста конкатенація) тексти  $A_i$  певної довжини, такої, що значно перевищує середню довжину повідомлення архіву. Кількість таких текстів залежить від розміру архіву, середньої довжини повідомлення та необхідної точності аналізу (еквівалентне затраті ресурсів);
- 2) Створюються архівовані копії текстів, згенерованих в першому кроці;
- 3) Повідомлення  $X$  додається до кожного оригінального тексту та архівуються способом, використаним в другому кроці;
- 4) Обраховується різниця між довжиною архівованого тексту  $A_i$  та тексту, отриманого в попередньому кроці;
- 5) Визначається коефіцієнт стиснення для повідомлення  $X$ , тобто відношення різниці довжин, отриманої в кроці 4, до довжини повідомлення  $X$ .

Попередньо аналогічна процедура повинна бути проведена для кожного повідомлення архіву, для визначення коефіцієнта стиснення для даного автора для усіх повідомлень. Усереднивши коефіцієнт стиснення можна сформулювати нижню межу його значення, перехід якою дозволяє вважати повідомлення таким, що належить цьому автору. Це підтверджується тим, що найбільшого коефіцієнту стиснення можна досягти тільки при використанні оптимального кодування. Якщо ж текст написаний іншим автором, завдяки стильовим особливостям, розподіл символів і їх послідовностей буде змінюватись, а отже дерево кодів Хаффмана, сформоване під час архівування текстів  $A_i$ , вже не буде оптимальним.

## Висновки

Метод, представлений у даній роботі, має широку сферу застосування – від простого розпізнавання мови до кластеризації масивів даних за ознаками

авторства, стилю, тематики. Завдяки використанню перевірених і максимально ефективних алгоритмів архівування процес аналізу є легким і швидким. Це дозволяє його використання у системах моментального аналізу даних, таких як аналізатори спаму. Завдяки відсутності необхідності аналізувати сам сенс повідомлення, метод може бути використаний для будь-яких даних, навіть не обов'язково смислових текстів.

### Перелік використаних джерел

1. Jacob Ziv, Abraham Lempel A Universal Algorithm for Sequential Data Compression // IEEE Transactions on Information Theory — 1997. — 23(3). — 337–343 с.
2. Сжатие по алгоритму Хаффмана — [algotist.manual.ru/compress/standard/huffman.php](http://algotist.manual.ru/compress/standard/huffman.php)
3. RFC 1951. DEFLATE Compressed Data Format Specification. — 1996. — 17 с.
4. Dario Benedetto. Language Trees and Zipping // Dario Benedetto, Emanuele Caglioti, Vittorio Loreto // Physical review letter— 2002. — 88(4). — 4 с.

1. Jacob Ziv, Abraham Lempel A Universal Algorithm for Sequential Data Compression // IEEE Trans-